

The image is a large, symmetrical, abstract graphic composed of the letters 'S' and 'Y' arranged in a grid-like pattern. The overall shape is a stylized 'Y' or a complex letterform. The top part is a wide horizontal bar made of 'S's, with 'Y's forming a central vertical stem. The sides of the 'Y' are also formed by 'S's and 'Y's, creating a sense of depth and structure. The letters are black on a white background, and the arrangement is highly regular and repetitive, suggesting a digital or algorithmic origin.

[illegible]

(5) 233
(6) 484
(7) 532
(8) 737
(9) 960
(10) 1125
(16) 1564
(19) 1720

DATA DECLARATIONS
\$GETCHN - Get Channel Information
\$GETDEV - Get Device Information
\$GETDVI - Get Device Information
DVI_DO_ITEM - Validate and move desired item
Special Items
Dual path and shadow set items
Get UCB from channel or device name


```
0000 1      .TITLE SYSGETDVI - System Services to Get Device Information
0000 2      .IDENT 'V04-000'
0000 3
0000 4
0000 5 *****
0000 6
0000 7      * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8      * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9      * ALL RIGHTS RESERVED.
0000 10
0000 11      * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12      * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13      * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14      * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15      * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16      * TRANSFERRED.
0000 17
0000 18      * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19      * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20      * CORPORATION.
0000 21
0000 22      * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23      * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24
0000 25 *****
0000 26
0000 27
0000 28      AUTHOR: Peter H. Lipman, CREATION DATE: 20-Oct-1981
0000 29
0000 30      MODIFIED BY:
0000 31
0000 32      V03-022 CWH3022 CW Hobbs 25-Jul-1984
0000 33      Change special items for shadow set information to call
0000 34      loadable support routine in mount verification (sysloa).
0000 35      This makes V4.n support of shadowing much simpler.
0000 36
0000 37      V03-021 LY0512 Larry Yetto 20-JUL-1984 13:01
0000 38      Fix bug in MEDIA_NAME code
0000 39
0000 40      V03-020 LY0502 Larry Yetto 10-JUL-1984 10:12
0000 41      Add support for the MEDIA_NAME and MEDIA_TYPE item codes
0000 42
0000 43      V03-019 TMK0001 Todd M. Katz 29-Apr-1984
0000 44      Add support for full length ( i.e. - LNMSC_NAMLENGTH )
0000 45      logical volume names. This support is accomplished through
0000 46      the following changes:
0000 47
0000 48      1. Change the scratch storage area within the local stack
0000 49      storage from LOG$C_NAMLENGTH to 4 bytes. This scratch area
0000 50      will now be used only to temporarily store values up to a
0000 51      longword in size.
0000 52
0000 53      2. Previously this scratch storage area had also been used to
0000 54      temporarily store character strings. Now, whenever a string
0000 55      must temporarily be stored, a KRP is used to provide the
0000 56      storage space. The KRP is allocated from the lookaside list
0000 57      the first time temporary storage is required for a character
```

0000 58 :
0000 59 :
0000 60 :
0000 61 :
0000 62 :
0000 63 :
0000 64 :
0000 65 :
0000 66 :
0000 67 :
0000 68 :
0000 69 :
0000 70 :
0000 71 :
0000 72 :
0000 73 :
0000 74 :
0000 75 :
0000 76 :
0000 77 :
0000 78 :
0000 79 :
0000 80 :
0000 81 :
0000 82 :
0000 83 :
0000 84 :
0000 85 :
0000 86 :
0000 87 :
0000 88 :
0000 89 :
0000 90 :
0000 91 :
0000 92 :
0000 93 :
0000 94 :
0000 95 :
0000 96 :
0000 97 :
0000 98 :
0000 99 :
0000 100 :
0000 101 :
0000 102 :
0000 103 :
0000 104 :
0000 105 :
0000 106 :
0000 107 :
0000 108 :
0000 109 :
0000 110 :
0000 111 :
0000 112 :
0000 113 :
0000 114 :

string. It remains allocated, and maybe utilized for temporarily storing other character strings, for the remainder of the current \$GETDVI invocation at which time it is returned to the KRP lookaside list.

- V03-018 RKS0018 RICK SPITZ 11-APR-1984
Enhance DVI_USE_DEVNAM to redirect references from a physical terminal UCB to the associated logical UCB.
- V03-017 LMP0221 L. Mark Pilant, 30-Mar-1984 16:35
Change UCBSL_OWNUIC to ORBSL_OWNER and UCBSW_VPROT to ORBSW_PROT.
- V03-016 MHB0115 Mark Bramhall 20-Mar-1984
Check for network device in SPC_TT_PHYDEVNAM.
- V03-015 MHB0104 Mark Bramhall 1-Mar-1984
Added SPC_TT_PHYDEVNAM for DVI\$_TT_PHYDEVNAM.
- V03-014 CWH3014 CW Hobbs 28-Feb-1984
Fix accvio when DVI\$_VOLSETMEM item is directed at a non-mounted device. Add special routine to get DVI\$_FREEBLOCKS for XQP disks, and several new routines for dual-path devices and shadow sets.
- V03-013 HH0002 Hai Huang 01-Feb-1984
Add job-wide mount support.
- V03-012 TCM0002 Trudy C. Matthews 04-Jan-1984
Document relationship between invocations of DVI_ITEM_CODE and DVI\$_xxx item codes defined by \$DVIDEF. Add warning to DVI_ITEM_CODE if this relationship is not preserved.
- V03-011 KFH0006 Ken Henderson 9 Sep 1983
Add documentation about adding itemcodes.
Add SPC_DEVLOCKNAM, SPC_VOLSETMEM.
- V03-010 KFH0005 Ken Henderson 30 Jul 1983
Removed debugging definitions
- V03-009 TCM0001 Trudy C. Matthews 24-Jun-1983
Add SPC_ALLDEVNAM:
- V03-008 DMW4040 DMWalp 31-May-1983
Intergrate new logical name structures.
- V03-007 KFH0004 Ken Henderson 18 May 1983
Changed SPC_FULLDEVNAM to new spec.
Added HEXSTR datatype to macro.
- V03-006 KFH0003 Ken Henderson 29 Apr 1983
Added SPC_FULLDEVNAM:
- V03-005 KFH0002 Ken Henderson 11 Mar 1983
Made .WARN for undefined item-codes more specific.

0000	115	:	V03-004	CWH1002	CW Hobbs	1-Mar-1983
0000	116	:		Add special item routines for DVI\$_PID and DVI\$_ACPPID		
0000	117	:				
0000	118	:	V03-003	KFH0001	Ken Henderson	23 Feb 1983
0000	119	:		Changed name of module to SYSGETDVI.		
0000	120	:		Added DVI_ITEM_CODE macro to replace ITEM		
0000	121	:		and SPECIAL macros. Moved tables and code		
0000	122	:		YF\$\$SYSGETDVI psect.		
0000	123	:				
0000	124	:	V03-002	PHL0103	Peter H. Lipman	24-Jul-1982
0000	125	:		Fix return status in IOSB to be \$\$\$_CONCEALED if device		
0000	126	:		was concealed. It was always being returned as \$\$\$_NORMAL		
0000	127	:		unless their was an error.		
0000	128	:				
0000	129	:	V03-001	KDM0002	Kathleen D. Morse	28-Jun-1982
0000	130	:		Added \$DEVDEF, \$PCBDEF, \$PRDEF, and \$PSLDEF. Fix comment.		
0000	131	:				
0000	132	:				
0000	133	: **				

```

0000 135 :      DEVO's GUIDE TO GETJPI/GETSYI/GETDVI
0000 136 :      -----
0000 137 :
0000 138 : Overview
0000 139 : -----
0000 140 :
0000 141 : These three system services are table-driven. The macro definition files
0000 142 : that help define their tables are shared with DCL and the RTL. This results
0000 143 : in new item-codes becoming useable with DCL's FSGETXXI lexical functions and
0000 144 : the RTL's LIB$GETXXI routines automatically. Additionally, new SYSBOOT
0000 145 : parameters become item-codes to the GETSYIs.
0000 146 :
0000 147 : The macro definition files are called JPITABLE.MAR, SYITABLE.MAR, and
0000 148 : DVITABLE.MAR, and live in MASDS:<VMSLIB.SRC>. During a systembuild, they
0000 149 : are inserted into the library SYSS$LIBRARY:SYSBLDMLB.MLB. DCL and the RTL
0000 150 : and SYS use this library to define their GETXXI tables. The system
0000 151 : parameter file <SYS.SRC>SYSPARAM.MAR has also been conditionalized to be
0000 152 : used to define GETSYI item-codes and is also inserted into SYSBLDMLB.MLB.
0000 153 :
0000 154 :
0000 155 : NOTE: SYSBLDMLB.MLB is a general macro library for holding macro
0000 156 : definitions that are shared between facilities, but will not
0000 157 : ship to the customer.
0000 158 :
0000 159 :
0000 160 : When adding an item-code, at least two files need to be edited. One of the
0000 161 : macro files listed above, as well as an SDL file that defines the 16-bit
0000 162 : number which is the user-visible item-code. Also, if a SYSBOOT parameter is
0000 163 : added, an SDL file needs to be updated to define the new GETSYI item-code.
0000 164 :
0000 165 : The GETDVI service actually uses only one table, but the GETSYI and GETJPI
0000 166 : services use several. The JPITABLE file defines all the tables for GETJPI
0000 167 : and the SYITABLE file defines all the tables for GETSYI. The different
0000 168 : tables group the peices of data according to method of retrieval.
0000 169 :
0000 170 : In some cases, the peice of data to be returned by the service requires
0000 171 : special processing to fetch, calculate, or format it before returning it.
0000 172 : In these cases, the code of the system service needs to be enhanced.
0000 173 : And if the data returned is a new format for DCL, the lexical function
0000 174 : module of DCL may need to be enhanced as well. Possibly the RTL code may
0000 175 : need enhancing as well.

```



```

0000 177
0000 178 :The Macros
0000 179 :-----
0000 180
0000 181 :A two-level scheme exists for defining the item tables used by the three
0000 182 :services and the other facilities. A commonly defined macro (called
0000 183 :JPI GENERATE TABLE, SYI GENERATE TABLE, or DVI GENERATE TABLE) contains
0000 184 :multiple calls to a lower-level macro (called JPI_ITEM_CODE, SYI_ITEM_CODE,
0000 185 :or DVI_ITEM_CODE) which actually defines each element in the table.
0000 186 :While the GENERATE TABLE macros are commonly defined, the ITEM_CODE macros
0000 187 :are individually defined according to the needs of facility. (For instance,
0000 188 :the LEXICON module must store the name of the item as an ASCII string - in
0000 189 :order to match it with the string supplied in the F$GETXXI function call;
0000 190 :the other facilities need not store the item name in text.)
0000 191
0000 192 :When an item-code must be added, an additional call to the ITEM_CODE macro
0000 193 :must be added to the appropriate GENERATE TABLE macro. In the case of GETJPI
0000 194 :and GETDVI, the GENERATE TABLE macro is defined in the JPITABLE and DVITABLE
0000 195 :modules. For GETDVI, an item-code definition must also be added to $DVIDEF.
0000 196 :BE SURE THAT THE ORDER OF THE ITEM CODE DEFINITIONS IN $DVIDEF IS THE SAME AS
0000 197 :THE ORDER OF INVOCATIONS OF DVI_ITEM_CODE IN DVITABLE. The item-code number
0000 198 :generated by $DVIDEF will be used as an index into DVI_ITEM_TABLE to locate
0000 199 :the appropriate information about that item.
0000 200
0000 201 :The SYI GENERATE TABLE macro is defined by the SYSPARAM module
0000 202 :- all the calls to the PARAMETER and PQL macros are 'collected' into the
0000 203 :SYI_GENERATE TABLE macro. When used in that mode (when GETSYSIW is defined),
0000 204 :the SYI_ITEMTABLES macro also becomes part of the SYI GENERATE TABLE macro.
0000 205 :SYI_ITEMTABLES is defined in the SYITABLE module and contains all the calls
0000 206 :to the SYI_ITEM_CODE macro that are Not related to SYSBOOT parameters.
0000 207 :When GETSYSIW is defined in SYSPARAM, the PARAMETER macro does not allocate
0000 208 :or store memory, but rather passes some of the arguments to it on through via
0000 209 :a call to SYI_ITEM_CODE. That is how all the calls to PARAMETER become calls
0000 210 :to SYI_ITEM_CODE.
0000 211
0000 212 :The following is the situation that exists when the symbol GETSYSIW is defined.
0000 213 :The non-SYSBOOT items are defined by the macro SYI_ITEMTABLES in SYITABLE.MAR.
0000 214 :The SYSBOOT items are defined by each invocation of the PARAMETER macro in
0000 215 :SYSPARAM.MAR. Note that each invocation of the PQL macro in SYSPARAM.MAR
0000 216 :invokes the PARAMETER macro twice. When GETSYSIW is defined, the PARAMETER
0000 217 :macro merely passes its arguments through to a call to the SYI_ITEM_CODE
0000 218 :macro. The SYI_ITEM_CODE macro is locally defined as needed by the facility.
0000 219
0000 220
0000 221
0000 222
0000 223
0000 224
0000 225
0000 226
0000 227
0000 228
0000 229
0000 230
0000 231

```



```
0000 233      .SBTTL DATA DECLARATIONS
0000 234
0000 235      :
0000 236      : System Services to Get Device Information
0000 237      :
0000 238      : $GETDEV and $GETCHN are obsolete and frozen starting with V3
0000 239      : $GETDVI replaces them and all new items are only available
0000 240      : via this system service.
0000 241      :
0000 242      :
0000 243      :
0000 244      : MACRO LIBRARY CALLS
0000 245      :
0000 246      :
0000 247      : $AQBDEF      : Define AQB offsets
0000 248      : $CCBDEF      : Define CCB offsets
0000 249      : $CDDDBDEF     : Define CDDDB offsets
0000 250      : $DCDEF      : Define adapter type codes
0000 251      : $DDBDEF      : Define DDB offsets
0000 252      : $DEVDEF      : Define device type codes
0000 253      : $DIBDEF      : Define DIB offsets
0000 254      : $DVIDEF      : Define Device/Volume Information constants
0000 255      : $JIBDEF      : Define Job Information Block offsets
0000 256      : $LKIDEF      : Define Get lock info codes
0000 257      : $LNMSTRDEF   : Define logical name structure offsets
0000 258      : $MTLDEF      : Define Mount List entry offsets
0000 259      : $ORBDEF      : Define Object's Rights Block offsets
0000 260      : $PCBDEF      : Define Process Control Block offsets
0000 261      : $PRDEF      : Define Processor Register numbers
0000 262      : $PSLDEF      : Define Program Status Longword fields
0000 263      : $RVTDEF      : Define RVT offsets
0000 264      : $SBDEF      : Define SB offsets
0000 265      : $SSDEF      : Define system status values
0000 266      : $TTDEF      : Define terminal DEVDEPEND bits
0000 267      : $TT2DEF     : Define terminal DEVDEPN2 bits
0000 268      : $TTYUCBDEF   : Define terminal UCB offsets
0000 269      : $UCBDEF      : Define UCB offsets
0000 270      : $VCBDEF      : Define VCB offsets
0000 271
0000 272      :
0000 273      : LOCAL MACROS
0000 274      :
0000 275      : Generate device information control table - $GETDEV and $GETCHN only
0000 276      :
0000 277      :
0000 278      : .MACRO GENTAB OFFSET,LENGTH
0000 279      : .BYTE LENGTH
0000 280      : .ENDM GENTAB
0000 281
0000 282      :
0000 283      : Generate field definitions for item value long word
0000 284      :
0000 285      : .MACRO DVIBITS NAME,SIZE
0000 286      : DVI_V 'NAME' = DVI_BIT
0000 287      : DVI_S 'NAME' = SIZE
0000 288      : DVI_BIT = DVI_BIT + SIZE
0000 289      : .ENDM DVIBITS
```

```
0000 290
0000 291 :
0000 292 : Generate the item-code table
0000 293 :
0000 294 .MACRO DVI_ITEM_CODE NAME,- : of the item-code
0000 295 SPECIAL,- : flag
0000 296 SOURCE,- : of the data
0000 297 DTYPE,- : of returned value
0000 298 BITPOS,- : of bitfield data
0000 299 OUTLEN,- : of returned value
0000 300 STRUCT,- : where the data lives
0000 301 DEVTYP : flag
0000 302
0000 303 .IF NOT_DEFINED DVI$_NAME
0000 304 .WARN ; DVI$_NAME IS NOT DEFINED IN STARDEFAE.SDL
0000 305 .IF_FALSE
0000 306 .IF NE ITEM_CODE-DVI$_NAME
0000 307 .WARN ; DEFINITION FOR ITEM CODE 'NAME IS OUT OF ORDER
0000 308 .ENDC
0000 309 .ENDC
0000 310
0000 311 ITEM_CODE = ITEM_CODE + 2
0000 312
0000 313 .IF IDENTICAL <SPECIAL><T>
0000 314
0000 315 .ADDRESS SPC_'NAME'
0000 316
0000 317 .IF_FALSE ; IDENTICAL <SPECIAL><T>
0000 318
0000 319 .IF DIFFERENT <SPECIAL><F>
0000 320 .WARN ; ERROR INVOKING DVI_ITEM_CODE FOR DVI$_NAME
0000 321 .ENDC ; DIFFERENT <SPECIAL><F>
0000 322
0000 323 XTYPE = DVI_C_VALUE
0000 324 .IIF IDENTICAL <DTYPE><HEXNUM>, XTYPE = DVI_C_VALUE
0000 325 .IIF IDENTICAL <DTYPE><DECNUM>, XTYPE = DVI_C_VALUE
0000 326 .IIF IDENTICAL <DTYPE><PRVMSK>, XTYPE = DVI_C_VALUE
0000 327 .IIF IDENTICAL <DTYPE><PRTMSK>, XTYPE = DVI_C_VALUE
0000 328 .IIF IDENTICAL <DTYPE><HEXSTR>, XTYPE = DVI_C_VALUE
0000 329 .IIF IDENTICAL <DTYPE><PADSTR>, XTYPE = DVI_C_VALUE
0000 330 .IIF IDENTICAL <DTYPE><CNTSTR>, XTYPE = DVI_C_CSTRING
0000 331 .IIF IDENTICAL <DTYPE><STRDSC>, XTYPE = DVI_C_VALUE
0000 332 .IIF IDENTICAL <DTYPE><BITVEC>, XTYPE = DVI_C_VALUE
0000 333 .IIF IDENTICAL <DTYPE><BITVAL>, XTYPE = DVI_C_BOOLEAN
0000 334 .IIF IDENTICAL <DTYPE><STDUIC>, XTYPE = DVI_C_VALUE
0000 335 .IIF IDENTICAL <DTYPE><STDTIM>, XTYPE = DVI_C_VALUE
0000 336 .IIF IDENTICAL <DTYPE><ACPTYP>, XTYPE = DVI_C_VALUE
0000 337
0000 338 .IF IDENTICAL <STRUCT><RVT>
0000 339 OFFVAL = DVI_C_'SOURCE'
0000 340 .IF_FALSE
0000 341 OFFVAL = 'STRUCT'$'SOURCE'
0000 342 .ENDC
0000 343
0000 344 .LONG <OFFVAL@DVI_V_OFFSET> ! -
0000 345 <'OUTLEN'@DVI_V_BYTCNT> ! -
0000 346 <DVI_C_'STRUCT'@DVI_V_STRUCT> ! -
```

```
0000 347      <XTYPE@DVI_V_DATATYPE> ! -
0000 348      <DVI_C 'DEVTYPE'@DVI_V_DEVTYPE> ! -
0000 349      <'BITPOS'@DVI_V_POSIT>
0000 350
0000 351      .ENDC ; IF_FALSE IDENTICAL <SPECIAL><T>
0000 352
0000 353      .ENDM DVI_ITEM_CODE
0000 354
0000 355      :
0000 356      : LOCAL SYMBOLS
0000 357      :
0000 358      : $GETDEV, $GETCHN Argument List Offset Definitions
0000 359      :
0000 360
00000004 0000 361      CHAN_DEVNAM=4 ; I/O channel number
0000 362      ; Device name descriptor
00000008 0000 363      PRILEN=8 ; Address to store length of primary string
0000000C 0000 364      PRIBUF=12 ; Address of primary buffer descriptor
00000010 0000 365      SCDLEN=16 ; Address to store length of secondary string
00000014 0000 366      SCDBUF=20 ; Address of secondary buffer descriptor
0000 367      :
0000 368      : Bit Field Definitions for Item Value long word
0000 369      :
00000000 0000 370      DVI_BIT = 0
0000 371      DVI_BITS_OFFSET,10 ; Offset in specified data structure
0000 372      DVI_BITS_BYTCNT,9 ; Size of item in bytes
0000 373      DVI_BITS_STRUCT,3 ; Structure (UCB, VCB)
0000 374      DVI_BITS_DATATYPE,3 ; Type of data item
0000 375      DVI_BITS_DEVTYPE,1 ; Device to which item is specific
0000 376      DVI_BITS_POSIT,5 ; Bit position of BITVAL dtype
0000 377      DVI_BITS_SPCFLG,1 ; THIS BIT MUST BE BIT 31!!!
0000 378      :
0000 379      : Datatype symbols for $GETDVI
0000 380      :
00000000 0000 381      DVI_C_VALUE = 0 ; Binary Value
00000001 0000 382      DVI_C_CSTRING = 1 ; Counted String
00000002 0000 383      DVI_C_BOOLEAN = 2 ; Bit value
0000 384      :
0000 385      : Mount type codes for SPC_DEVLOCKNAM
0000 386      :
00000001 0000 387      DVI_K_PRIVATE = 1
00000002 0000 388      DVI_K_SHAREABLE = 2
0000 389      :
0000 390      : Structure code symbols for $GETDVI
0000 391      :
0000 392      :
00000000 0000 393      DVI_C_UCB = 0 ; Unit Control Block
00000001 0000 394      DVI_C_DDB = 1 ; Device Data Block
00000002 0000 395      DVI_C_VCB = 2 ; Volume Control Block
00000003 0000 396      DVI_C_RVT = 3 ; Relative Volume Table
00000004 0000 397      DVI_C_AQB = 4 ; ACP Queue Header Block
00000005 0000 398      DVI_C_ORB = 5 ; Object's Rights Block
0000 399      :
0000 400      : Device type codes for $GETDVI
0000 401      :
00000000 0000 402      DVI_C_ANY = 0 ; Any device
00000001 0000 403      DVI_C_DISK = 1 ; Disk only
```



```
0000 404 :: Relative Volume Table Item Sub Codes for $GETDVI - in OFFSET field
0000 405 ::
0000 406 ::
00000000 0000 407 DVI_C_VOLCOUNT = 0 ; Count of volumes in volume set
00000001 0000 408 DVI_C_ROOTDEVNAM = 1 ; Device name for first volume in vol set
00000002 0000 409 DVI_C_NEXTDEVNAM = 2 ; Device name for next volume in vol set
0000 410 ::
0000 411 :: Local Storage Offsets
0000 412 ::
0000 413 $OFFSET 0,NEGATIVE,<-
0000 414 <PRIMARY_UCB,16>,- ; Primary UCB/VCB, Secondary UCB/VCB
0000 415 <CURRENT_UCB,8>,- ; Current UCB/VCB
0000 416 RETLEN_ADR,- ; Address to return length
0000 417 SCRATCH,- ; Scratch storage - 4 bytes ONLY
0000 418 KRP,- ; Address of allocated KRP
0000 419 <SCRATCH_SIZE,0>,- ; Size of local storage
0000 420 STATUS,- ; Returned Success Status
0000 421 SAVED_ASTADR,- ; Saved ASTADR parameter
0000 422 IOUNLOCK,- ; Need to unlock I/O data base if LBS
0000 423 >
FFFO PRIMARY_UCB:
FFE8 CURRENT_UCB:
FFE4 RETLEN_ADR:
FFE0 SCRATCH:
FFDC KRP:
FFDC SCRATCH_SIZE:
FFD8 STATUS:
FFD4 SAVED_ASTADR:
FFD0 IOUNLOCK:
FFFFFFDA 0000 424 RETLEN = STATUS+2 ; Return Length $GETDEV, $GETCHN
FFFFFFF4 0000 425 PRIMARY_VCB = PRIMARY_UCB+4 ; Primary VCB address
FFFFFFF8 0000 426 SECONDARY_VCB = PRIMARY_UCB+8 ; Secondary VCB address
FFFFFFFC 0000 427 SECONDARY_VCB = PRIMARY_UCB+12 ; Secondary VCB address
FFFFFFEC 0000 428 CURRENT_VCB = CURRENT_UCB+4 ; Current VCB address
0000 429 ::
0000 430 :: The following ASSUMES guarantee the consistency of the ACP type
0000 431 :: definition in $AQBDEF and the user visible constants in $DVIDEF
0000 432 ::
0000 433 ASSUME AQB$K_F11V1 EQ DVISC_ACP_F11V1 ; FILES-11 STRUCTURE LEVEL 1
0000 434 ASSUME AQB$K_F11V2 EQ DVISC_ACP_F11V2 ; FILES-11 STRUCTURE LEVEL 2
0000 435 ASSUME AQB$K_MTA EQ DVISC_ACP_MTA ; MAGTAPE
0000 436 ASSUME AQB$K_NET EQ DVISC_ACP_NET ; NETWORKS
0000 437 ASSUME AQB$K_REM EQ DVISC_ACP_REM ; REMOTE I/O
0000 438 ASSUME AQB$K_JNL EQ DVISC_ACP_JNL ; JOURNAL
0000 439 ::
0000 440 :: LOCAL DATA
0000 441 ::
0000 442 ::
0000 443 :: Device Information Control Table - $GETDEV, $GETCHN
0000 444 ::
0000 445 ::
00000000 0000 446 .PSECT YF$$$SYSGETDVI
0000 447 DEVTAB:
0000 448 GENTAB L_DEVCHAR,4 ; Device characteristics
0001 449 GENTAB B_DEVCLASS,1 ; DEVCLASS - Device Class
0002 450 GENTAB B_DEVTYPE,1 ; DEVTYPE - Device Type
0003 451 GENTAB W_DEVBUFSIZ,2 ; DEVBUFSIZ - buffer size
```

```
0004 452 GENTAB L_DEVDEPEND,4 ; DEVDEPEND - device dependent info
0005 453 GENTAB W_UNIT,<2+2> ; Device unit number
0006 454 ; DIBSW_DEVNAMOFF <-- 0
0006 455 GENTAB L_PID,4 ; Device owner process identification
0007 456 GENTAB L_OWNUIC,4 ; Device owner user identification code
0008 457 GENTAB W_VPROT,2 ; Device protection mask
0009 458 GENTAB W_ERRCNT,2 ; Device error count
000A 459 GENTAB L_OPCNT,<4+2> ; Device operations complete count
000B 460 ; DIBSW_VOLNAMOFF <-- 0
000B 461 GENTAB W_RECORDSZ,2 ; Blocked Record Size
B4 000C 462 .BYTE -DIBSL_MAXBLOCK+DIBST_DEVNAME ; Skip over string area
000D 463 GENTAB L_MAXBLOCK,4 ; Disk size in blocks
00 000E 464 .BYTE 0 ; End of table
000F 465 :
000F 466 :*****
000F 467 :
000F 468 : GENERATE THE ITEM-CODE TABLE
000F 469 :
000F 470 :*****
000F 471 :
000F 472 DVI_ITEM_TABLE:
000F 473
000F 474 ; Index 0 is not used by EXESGETDVI
000F 475 :
00000000 000F 476 .LONG 0
00000002 0013 477
00000002 0013 478 ITEM_CODE = 2
00000002 0013 479
00000002 0013 480 DVI_GENERATE_TABLE
025F 481
00000127 025F 482 MAX_ITEM_CODE = <<.-DVI_ITEM_TABLE>/2>-1
```

```

025F 484 .SBTTL $GETCHN - Get Channel Information
025F 485
025F 486 :+ EXE$GETCHN - Get channel information
025F 487
025F 488 This service provides the capability to retrieve information about a
025F 489 device that is assigned to a channel and its associated device if any.
025F 490
025F 491 INPUTS:
025F 492
025F 493     CHAN(AP) = I/O channel number.
025F 494     PRILEN(AP) = Address to store length of primary device information.
025F 495     PRIBUF(AP) = Address of primary buffer descriptor.
025F 496     SCDLEN(AP) = Address to store length of secondary device information.
025F 497     SCDBUF(AP) = Address of secondary buffer descriptor.
025F 498
025F 499     R4 = Current process PCB address.
025F 500
025F 501 OUTPUTS:
025F 502
025F 503     R0 low bit clear indicates failure to retrieve device information.
025F 504
025F 505         R0 = SSS_ACCVIO - primary or secondary buffer descriptor
025F 506             cannot be read by calling access mode, or primary
025F 507             buffer, primary buffer length, secondary buffer, or
025F 508             secondary buffer length cannot be written by calling
025F 509             access mode.
025F 510
025F 511         R0 = SSS_IVCHAN - invalid channel number specified.
025F 512
025F 513         R0 = SSS_NOPRIV - specified channel is not assigned to a
025F 514             device or the calling access mode does not have
025F 515             privilege to access the channel.
025F 516
025F 517     R0 low bit set indicates successful completion.
025F 518
025F 519         R0 = SSS_BUFFEROVF - normal completion, all characteristic
025F 520             information did not fit in specified buffer(s).
025F 521
025F 522         R0 = SSS_NORMAL - normal completion, all characteristic
025F 523             information transferred.
025F 524
025F 525 :-
00000000 526 .PSECT Y$EXEPAGED
0000 527
0000 528 .ENTRY EXE$GETCHN, *M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
51 098F'CF 9E 0002 529 MOVAB W^DVI USE CHAN,R1 ; Use channel parameter
0255' 31 0007 530 BRW EXE_GETDEV ; Join GETDEV code

```



```
000A 532 .SBTTL $GETDEV - Get Device Information
000A 533
000A 534 :+ EXE$GETDEV - Get device information
000A 535
000A 536 This service provides the capability to retrieve information about a
000A 537 device and its associated device if any.
000A 538
000A 539 INPUTS:
000A 540
000A 541 DEVNAM(AP) = Address of device name descriptor.
000A 542 PRILEN(AP) = Address to store length of primary device information.
000A 543 PRIBUF(AP) = Address of primary buffer descriptor.
000A 544 SCLEN(AP) = Address to store length of secondary device information.
000A 545 SCDBUF(AP) = Address of secondary buffer descriptor.
000A 546
000A 547 R4 = Current process PCB address.
000A 548
000A 549 OUTPUTS:
000A 550
000A 551 R0 low bit clear indicates failure to retrieve device information.
000A 552
000A 553 R0 = SS$ ACCVIO - Device name string, device name string
000A 554 descriptor, primary buffer descriptor, or secondary
000A 555 buffer descriptor cannot be read by calling access
000A 556 mode, or primary buffer, primary buffer length,
000A 557 secondary buffer, or secondary buffer length cannot
000A 558 be written by calling access mode.
000A 559
000A 560 R0 = SS$ IVDEVNAM - Device name string contains invalid
000A 561 characters, or no device device name string descriptor
000A 562 specified.
000A 563
000A 564 R0 = SS$ IVLOGNAM - Zero or greater than maximum length device
000A 565 name string specified.
000A 566
000A 567 R0 = SS$ _NONLOCAL - Device exists on a remote system.
000A 568
000A 569 R0 = SS$ _NOSUCHDEV - Specified device does not exist on host
000A 570 system.
000A 571
000A 572 R0 low bit set indicates successful completion.
000A 573
000A 574 R0 = SS$ BUFFEROVF - Normal completion, all characteristic
000A 575 information did not fit in specified buffer(s).
000A 576
000A 577 R0 = SS$ NORMAL - Normal completion, all characteristic
000A 578 information transferred.
000A 579 :-
000A 580
000A 581 .ENTRY EXE$GETDEV, ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
000C 582 MOVAB W^DVI USE DEVNAM,R1 ; Use device name descriptor
0011 583 BRW EXE_GETDEV
0014 584
0000 585 .PSECT YF$$SYSGETDVI
025F 586
025F 587 : R1 = address of USE_CHAN or USE_DEVNAM entry point
025F 588 :
```

51 09B3'CF 0FFC 9E 000C 582
024B' 31 0011 583
0014 584
0000 585
025F 586
025F 587
025F 588

```
025F 589 .ENABLE LSB
025F 590
025F 591 EXE_GETDEV:
025F 592 MOVAL SCRATCH_SIZE-4(SP),SP ; Reserve scratch storage
0263 593 ; include uninitialized return status
0263 594 CLRQ -(SP) ; Init SAVED_ASTADR and IOUNLOCK flag
0265 595
0265 596 ; The above stack locations are all referenced by offsets from FP
0265 597
0265 598 MOVL CHAN_DEVNAM(AP),R0 ; Get CHAN or DEVNAM parameter
0269 599 JSB (R1) ; Set up UCB's to be used
0268 600 BLBC R0,40$ ; Branch if error
026E 601 MOVL #SS$_NORMAL,STATUS(FP) ; Init normal success status
0272 602 ; Overwrite possible SS$_CONCEALED
0272 603 CLRL R9 ; Primary device items
0274 604 MOVQ PRILEN(AP),R7 ; Get primary buffer parameters
0278 605 BSBB FILBUF ; Fill primary buffer
027A 606 BLBC R0,40$ ; Branch if error
027D 607 MOVL #1,R9 ; Secondary device items
0280 608 MOVQ SCOLEN(AP),R7 ; Get secondary buffer parameters
0284 609 BSBB FILBUF ; Fill secondary buffer
0286 610 BLBC R0,40$ ; Branch if error
0289 611 30$: MOVZWL STATUS(FP),R0 ; Get normal or overflow status to return
028D 612 40$: BLBS IOUNLOCK(FP),50$ ; Branch if must unlock I/O data base
0291 613 RET
0292 614 50$: JMP G*IOCSUNLOCK ; Unlock I/O database and return
0298 615
0298 616 .DSABL LSB
0298 617
0298 618 ; Subroutine to fill characteristic buffer
0298 619
0298 620 INPUTS:
0298 621
0298 622 R7 = Address to return length of data stored
0298 623 R8 = Descriptor of DIB buffer
0298 624 R9 = 0 if getting primary characteristics
0298 625 = 1 if getting secondary characteristics
0298 626
0298 627 OUTPUTS:
0298 628
0298 629 R0 = Status
0298 630 R1 through R11 altered
0298 631
0298 632 ACCVIO_1:
0298 633
0298 634 BRW ACCVIO
0298 635 FILBUF: TSTL R8 ; Any buffer specified?
029D 636 BNEQ 5$ ; Branch if yes
029F 637 BRW 160$ ; No, nothing to do
02A2 638 5$: MOVQ PRIMARY_UCB(FP)[R9],CURRENT_UCB(FP) ; Set current UCB/VCB address
02A8 639 MOVL R7,RETLEN_ADR(FP) ; Save address for return length
02AC 640 IFNORD #8,(R8),ACCVIO_1 ; ACCVIO if cannot read out buf decriptor
02B2 641 MOVL 4(R8),R7 ; Get the address
02B6 642 MOVZWL (R8),R6 ; and the size of the buffer
02B9 643 ASSUME DIB$_LENGTH LE 512
02B9 644 CMPW R6,#DIB$_LENGTH ; If buffer is larger than needed
02BE 645 BGEQU 20$ ; then use the maximum size for probe
```

```
DB AD 0601 8F B0 02C0 646 MOVW #SS$_BUFFEROVF,STATUS(FP) ; Record buffer overflow status
      04 11 02C6 647 BRB 30$
      56 74 8F 9A 02C8 648 20$: MOVZBL #DIB$K_LENGTH,R6 ; Actual size of data to be returned
      DA AD 56 B0 02CC 649 30$: MOVW R6,RETLEN(FP) ; Remember how much data will be returned
      SB FD26 CF DE 02D0 650 IFNOWRT R6,(R7),ACCVIO ; Can entire buffer be written?
      58 D4 02D6 651 MOVAL W*DEVTAB,R11 ; Address of item lengths
      7E 56 7D 02DB 652 CLRL R8 ; No item return length
      56 DD 02DD 653 MOVQ R6,-(SP) ; Save DIB descriptor
      5A 88 98 02E0 654 PUSHL R6 ; Scratch copy of length
      07 14 02E2 655 40$: CVTBL (R11)+,R10 ; Length of buffer for next item
      1D 13 02E5 656 BGTR 50$ ; Branch if item to move
      5A 5A CE 02E7 657 BEQL 90$ ; Branch if end of table
      10 11 02E9 658 MNEGL R10,R10 ; Skip over section of DIB
      59 02 C0 02EE 659 BRB 70$
      6E 5A D1 02F1 660 50$: ADDL #2,R9 ; Next item code
      05 15 02F4 661 CMPL R10,(SP) ; Enough room for this item?
      5A 6E D0 02F6 662 BLEQ 60$ ; Branch if yes
      0B 15 02F9 663 MOVL (SP),R10 ; No, use what space is left
      018A 30 02FB 664 BLEQ 90$ ; All done if no space left
      57 5A C0 02FE 665 60$: BSBW DVI_DO_ITEM ; Put the next item in the DIB
      6E 5A C2 0301 666 70$: ADDL R10,R7 ; Next free location in DIB
      DC 14 0304 667 SUBL R10,(SP) ; Adjust space left in DIB
      0306 668 BGTR 40$ ; Branch if room for another item
      0306 669
      0306 670 ; The DIB is now filled in except for the device controller name string
      0306 671 ; and the volume name string and their respective offset locations.
      0306 672 ; DIB$W_DEVNAMEOFF and DIB$W_VOLNAMEOFF are currently 0. The string area
      0306 673 ; is deliberately NOT backgrounded so that no data is written except that
      0306 674 ; which is explicitly returned.
      0306 675
      00E0 8F BA 0306 676 90$: POPR #*M<R5,R6,R7> ; Clean off scratch cell,
      56 24 A2 030A 677 SUBW #DIB$T_DEVNAME,R6 ; recover DIB descriptor
      2B 15 030D 678 BLEQ 150$ ; Room for CTLNAM and VOLNAM string
      53 24 A7 DE 030F 679 MOVAL DIB$T_DEVNAME(R7),R3 ; Branch if no room for strings
      55 E8 AD D0 0313 680 MOVL CURRENT_UCB(FP),R5 ; Starting adr in DIB for strings
      28 A5 14 C1 0317 681 ADDL3 #DDB$T_NAME,UCB$L_DDB(R5),R5 ; Address of UCB
      54 85 9A 031C 682 MOVZBL (R5)+,R4 ; Address of ASCII controller name
      06 13 031F 683 BEQL 110$ ; Size in R4, adr in R5
      58 0E A7 DE 0321 684 MOVAL DIB$W_DEVNAMEOFF(R7),R8 ; Branch if controller name null
      2B 10 0325 685 BSBB MOVE_NAME ; Address to store offset to string
      55 EC AD D0 0327 686 110$: MOVW CURRENT_VCB(FP),R5 ; Move the name, set up the offset
      0D 13 032B 687 MOVL CURRENT_VCB(FP),R5 ; Address of VCB
      55 14 A5 DE 032D 688 BEQL 150$ ; Branch if volume not mounted
      54 0C D0 0331 689 MOVL VCB$T_VOLNAME(R5),R5 ; Adr of 12 byte blank filled volume name
      58 20 A7 DE 0334 690 MOVL #12,R4 ; Size of name string
      1B 10 0338 691 MOVAL DIB$W_VOLNAMEOFF(R7),R8 ; Address to store offset to string
      033A 692 BSBB MOVE_NAME ; Move the name, set up the offset
      033A 693
      033A 694 ; DIB is now totally filled in, return length to caller if requested
      033A 695
      50 E4 AD D0 033A 696 150$: MOVL RETLEN_ADR(FP),R0 ; Address to return DIB length
      0A 13 033E 697 BEQL 160$ ; Branch if none specified
      0340 698 IFNOWRT #2,(R0),ACCVIO ; Branch if length cannot be written
      60 DA AD B0 0346 699 MOVW RETLEN(FP),(R0) ; Return the DIB length
      50 01 3C 034A 700 160$: MOVZWL #SS$_NORMAL,R0 ; Set successful completion
      05 034D 701 RSB
      034E 702 ACCVIO:
```



```
50 0C 3C 034E 703      MOVZWL #SS$_ACCVIO,R0      : Access violation
      05 0351 704      RSB
      0352 705      :
      0352 706      : Move name string and fill in DIB offset to it
      0352 707      :
      0352 708      : INPUTS:
      0352 709      :
      0352 710      : R3 = Address to store data
      0352 711      : R4 = Byte count to store
      0352 712      : R5 = Source string to store
      0352 713      : R6 = Count of bytes remaining in output buffer
      0352 714      : R7 = Base address of DIB
      0352 715      : R8 = Address to store offset to string
      0352 716      :
      0352 717      : OUTPUTS:
      0352 718      :
      0352 719      : R3 = Updated address to store next string
      0352 720      : R6 = Updated space remaining to store next string
      0352 721      : R0 through R5 altered
      0352 722      : Other registers preserved
      0352 723      :
      0352 724      : MOVE_NAME:
      0352 725      :
      0354 726      : DECL R6      : Room for byte count for string
      0356 727      : BLSS 20$,R7,R3,R0 : Branch if not, don't store offset
      035A 728      : MOVW R0,(R8) : Offset to string
      035D 729      : MOVW R4,(R3)+ : Store offset in DIB
      0360 730      : CMPL R4,R6 : Store count for ASCII string
      0363 731      : BLEQ 10$,R4 : Enough room for rest of string?
      0365 732      : MOVW R6,R4 : Branch if yes
      0368 733 10$: : No, use what is left
      0368 734      : SUBL R4,R6 : Keep track of space remaining
      036F 735 20$: : Store the string
      036F 735      : RSB
```

```
0370 737 .SBTTL SGETDVI - Get Device Information
0370 738 ++
0370 739
0370 740 FUNCTIONAL DESCRIPTION:
0370 741
0370 742 This service allows a process to get information about a device
0370 743 it currently has a channel assigned to, or one it explicitly names.
0370 744
0370 745 CALLING SEQUENCE:
0370 746
0370 747 CALLS/CALLG
0370 748
0370 749 INPUTS:
0370 750
0370 751 EFN(AP) = number of the event flag to set when all of the requested
0370 752 data is valid.
0370 753 CHAN(AP) = channel to which desired device is assigned or 0
0370 754 if specifying device by name.
0370 755 DEVNAM(AP) = address of a string descriptor for the device name
0370 756 or logical device name desired. This is only used
0370 757 if the channel parameter is 0.
0370 758 ITMLST(AP) = address of a list of item descriptors of the form:
0370 759
0370 760 +-----+
0370 761 | ITEM CODE ! BUF. LENGTH !
0370 762 +-----+
0370 763 | BUFFER ADDRESS |
0370 764 +-----+
0370 765 | ADDRESS TO RETURN LENGTH |
0370 766 +-----+
0370 767
0370 768 IOSB(AP) = address of a quadword I/O status block to receive final
0370 769 status
0370 770 ASTADR(AP) = address of an AST routine to be called when all of the
0370 771 requested data has been supplied.
0370 772 ASTPRM(AP) = 32 bit ast parameter
0370 773 NULARG(AP) = Reserved argument - address of a buffer descriptor
0370 774 for wild device context.
0370 775
0370 776 R4 = Current process PCB address
0370 777
0370 778 IMPLICIT INPUTS:
0370 779
0370 780 none
0370 781
0370 782 OUTPUTS:
0370 783
0370 784 none
0370 785
0370 786 IMPLICIT OUTPUTS:
0370 787
0370 788 none
0370 789
0370 790 ROUTINE VALUE:
0370 791
0370 792 R0 low bit clear indicates failure to retrieve device information
0370 793
```

```
0370 794 : RO = $$$ACCPIO - Device name string descriptor, device
0370 795 : name string, or ITMLST cannot be read by the
0370 796 : calling access mode. Item buffer or return
0370 797 : length word cannot be written by the calling
0370 798 : access mode.
0370 799 :
0370 800 : RO = $$$IVCHAN - Invalid channel number specified
0370 801 :
0370 802 : RO = $$$IVDEVNAM - Device name string contains invalid
0370 803 : characters, or no device name string was
0370 804 : specified and no channel number was specified.
0370 805 :
0370 806 : RO = $$$IVLOGNAM - Zero or greater than maximum length
0370 807 : device name string specified.
0370 808 :
0370 809 : RO = $$$NONLOCAL - Device exists on a remote system
0370 810 :
0370 811 : RO = $$$NOSUCHDEV - Specified device does not exist on
0370 812 : host system
0370 813 :
0370 814 : RO = $$$BADPARAM - An invalid item identifier was specified
0370 815 :
0370 816 : RO = $$$EXASTLM - An AST was requested and the AST quota
0370 817 : was exceeded.
0370 818 :
0370 819 : RO low bit set indicates successful completion.
0370 820 :
0370 821 : RO = $$$NORMAL - Normal completion
0370 822 :
0370 823 : SIDE EFFECTS:
0370 824 :
0370 825 : none
0370 826 : --
0370 827 :
0370 828 : Equated Symbols:
0370 829 :
0370 830 : Argument List Offsets
0370 831 :
0370 832 :
0370 833 :
00000004 0370 834 : EFN = 4 ; Event flag number argument
00000008 0370 835 : CHAN = 8 ; Channel assigned to device or 0
0000000C 0370 836 : DEVNAM = 12 ; Address of device name string descriptor
00000010 0370 837 : ITMLST = 16 ; Address of item identifiers
00000014 0370 838 : IOSB = 20 ; I/O status block address
00000018 0370 839 : ASTADR = 24 ; AST routine address
0000001C 0370 840 : ASTPRM = 28 ; AST parameter
00000020 0370 841 : NULARG = 32 ; Reserved argument - wild context buf desc
0370 842 :
0370 843 :
00000014 0370 844 : .PSECT YSEXEPAGED
0014 845 :
0357' OFFC 0014 846 : .ENTRY EXE$GETDVI, ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
31 0016 847 : BRW EXE_GETDVI
0019 848 :
00000370 0370 849 : .PSECT YF$$$SYSGETDVI
0370 850 :
```



```
0370 851 .ENABL LSB
0370 852
0370 853 EXE_GETDVI:
0370 854     MOVAL   SCRATCH_SIZE(SP),SP      ; Allocate local storage
0374 855     CLRL    KRP(FP)                  ; Initially no KRP is allocated
0377 856     PUSHL   #SS$ NORMAL              ; Set presumed normal success status
0379 857     CLRQ    -(SP)                    ; Zero SAVED_ASTADR and IOUNLOCK
037B 858
037B 859 ; The above stack locations are all referenced by offsets from FP
037B 860 ;
037B 861     MOVZBL   EFN(AP),R3                ; Get event flag number
037F 862     JSB     G^SCH$CLREF              ; Clear this event flag
0385 863     BLBC     R0,DVI_ERROR              ; If error, exit with error status
038B 864     MOVL    IOSB(AP),R1              ; Get IOSB address if specified
038C 865     BEQL    10$                      ; Branch if none specified
038E 866     IFNOWRT #8,(R1),DVI_ACCVIO       ; If not writable by caller then ACCVIO
0394 867     CLRQ    (R1)                    ; Clear the IOSB
D4 AD 18 AC D0 0396 868 10$: MOVL    ASTADR(AP),SAVED_ASTADR(FP) ; Save ASTADR parameter
039B 869     BEQL    20$                      ; Branch if none specified
039D 870     TSTW    PCB$W ASTCNT(R4)         ; If AST limit is exceeded
03A0 871     BLEQ    DVI_EXASTLM              ; then indicate error
03A2 872
03A2 873 ; See if Channel parameter was specified
03A2 874 ;
03A2 875 20$: MOVZWL   CHAN(AP),R0              ; Fetch channel parameter if specified
03A6 876     BEQL    30$                      ; Branch if not specified
03A8 877     BSBW    DVI_USE_CHAN              ; Get UCB address from channel
03AB 878     BLBS     R0,40$                  ; Branch if no error
03AE 879 DVI_ERROR:
03AE 880     BRB       DVI_ERROR_1
03B0 881 DVI_ACCVIO:
03B0 882     MOVL     S^#SS$ ACCVIO,R0          ; Access violation
03B3 883     BRB       DVI_ERROR
03B5 884 DVI_EXASTLM:
03B5 885     MOVZWL   #SS$ EXASTLM,R0           ; Exceeded ASTLM quota
03BA 886     BRB       DVI_ERROR
03BC 887 DVI_BADPARAM:
03BC 888     MOVZWL   #SS$ BADPARAM,R0         ; Bad parameter
03BF 889     BRB       DVI_ERROR
03C1 890
03C1 891 ; Use Device Name String parameter to locate desired device
03C1 892 ;
03C1 893 30$: MOVL     DEVNAM(AP),R0              ; Get the device name descriptor
03C5 894     BSBW     DVI_USE_DEVNAM           ; Get UCB using device name
03C8 895     BLBC     R0,DVI_ERROR              ; Branch if error
03CB 896
03CB 897 ; I/O data base locked for reading
03CB 898 ;
03CB 899 40$: MOVL     ITMLST(AP),R11            ; Address of list of items
03CF 900     IFNORD   #4,(R11),DVI_ACCVIO      ; Check first long word readable
03D5 901 50$: MOVZWL   2(R11),R9              ; Item code for next item
03D9 902     BEQL     DVI_COMPLETE              ; Done if zero, take normal exit
03DB 903     CMPW     R9,#MAX_ITEM_CODE          ; Valid item code?
03E0 904     BGTRU    DVI_BADPARAM               ; Branch if not
03E2 905     CMPL     R9,#1                      ; 0 and 1 are not used
03E5 906     BLEQ     DVI_BADPARAM               ; Branch if bad item code
03E7 907     MOVAQ    (R11)+,R1                  ; R1 = R11 = Adr of item buf descriptor
```

```
00000000'GF 16 03EA 908 JSB G*EXESPROBEW_DSC ; R11 = R11 + 8
; See if caller can read buf dsc
; and write the buffer it describes
BB 50 E9 03F0 909 BLBC R0,DVI_ERROR ; Branch if not
57 52 D0 03F3 910 MOVL R2,R7 ; Save item buffer address
5A 51 3C 03F6 911 MOVZWL R1,R10 ; and its size
04 5A D1 03F9 912 CMPL R10,#4 ; Item buffer at least 4 bytes?
BE 19 03FC 913 BLSS DVI_BADPARAM ; Branch if not
03FE 914 IFNORD #8,(R11),DVI_ACCVIO ; Check access to rest of this item
; and first long word of the next
58 8B D0 0404 915 MOVL (R11)+,R8 ; R8 = Address to return length
06 13 0407 916 BEQL 60$ ; Branch if return length not requested
0409 917 IFNOWRT #2,(R8),DVI_ACCVIO ; Make sure caller can write this
50 59 01 00 EF 040F 920 EXTZV #0,#1,R9,R0 ; Get primary/secondary flag
EB AD F0 AD40 7D 0414 921 MOVQ PRIMARY_UCB(FP)[R0],CURRENT_UCB(FP) ; Set current UCB/VCB
006B 30 041A 922 BSBW DVI_DO_ITEM ; Process this item
B6 11 041D 923 BRB 50$ ; Get next item
041F 924 ;
041F 925 ; R0 = error status
041F 926 ;
041F 927 ;
DB AD 50 B0 041F 928 DVI_ERROR_1:
0423 929 MOVW R0,STATUS(FP) ; Save error status
0423 930 ;
0423 931 ; Normal completion, saved status already set
0423 932 ;
0423 933 DVI_COMPLETE:
54 00000000'9F D0 0423 934 MOVL @#CTL$GL_PCB,R4 ; Get PCB address
09 D0 AD E9 042A 935 BLBC IOUNLOCK(FP),70$ ; Branch if no read lock to release
00000000'GF 16 042E 936 JSB G*SCH$IOUNLOCK ; Unlock I/O data base
51 60 A4 D0 0434 937 SETIPL #0 ; Allow all interrupts
52 D4 043B 938 70$: MOVL PCBSL_PID(R4),R1 ; Get process's PID
53 04 AC D0 043D 939 CLRL R2 ; Set null priority increment
00000000'GF 16 0441 940 MOVL EFN(AP),R3 ; Get event flag number to set
51 14 AC D0 0447 941 JSB G*SCH$POSTEF ; Set the event flag
OA 13 044B 942 MOVL IOSB(AP),R1 ; Get address of IOSB
61 D8 AD B0 044D 943 BEQL 80$ ; Branch if none
55 D4 AD D0 0453 944 IFNOWRT #8,(R1),80$ ; Check if writable
15 13 045B 945 MOVW STATUS(FP),(R1) ; Store completion status
54 54 02 16 EF 045D 946 80$: MOVL SAVED_ASTADR(FP),R5 ; Get address of AST routine
0464 947 BEQL 90$ ; Branch if none specified
50 D8 AD 3C 0472 948 MOVPSL R4 ; Get PSL
57 DC AD D0 0476 949 EXTZV #PSL$V_PVPMOD,#PSL$S_PVPMOD,R4,R4 ; Extract previous mode
0B 13 047A 950 SDCLAST S (R5),ASTPRM(AP),R4 ; Queue the completion AST
04 B6 67 OE 0483 951 90$: MOVZWL STATUS(FP),R0 ; Return saved status
0487 952 ;
0488 953 MOVL KRP(FP),R7 ; Retrieve address of allocated KRP
9E 047C 954 BEQL 100$ ; Immediately return if no KRP allocated
04 0483 955 MOVAB G*CTL$GL_KRPFL,R6 ; Else retrieve address of KRP listhead
04 0487 956 INSQUE (R7),@4(R6) ; and deallocate KRP to lookaside list
0488 957 100$: RET ; Exit system service
0488 958 .DSABL LSB
```

```
0488 960 .SBTTL DVI_DO_ITEM - Validate and move desired item
0488 961
0488 962 :++
0488 963
0488 964 FUNCTIONAL DESCRIPTION:
0488 965
0488 966 Routine to validate item identifier and return the desired
0488 967 information to the caller's buffer.
0488 968
0488 969 CALLING SEQUENCE:
0488 970
0488 971 JSB/BSB
0488 972
0488 973 INPUTS:
0488 974
0488 975 R7 = Address of buffer to return item - already probed
0488 976 R8 = Address of buffer to return length - already probed
0488 977 0 if not returning length
0488 978 R9 = Item code
0488 979 R10 = Size of buffer for item
0488 980
0488 981 IMPLICIT INPUTS:
0488 982
0488 983 CURRENT_UCB(FP) - Address of the UCB
0488 984 CURRENT_VCB(FP) - Address of the VCB
0488 985 SCRATCH(FP) - 4 bytes of scratch storage
0488 986 KRP(FP) - Address of allocated KRP, if any
0488 987
0488 988 OUTPUTS:
0488 989
0488 990 none
0488 991
0488 992 IMPLICIT OUTPUTS:
0488 993
0488 994 KRP(FP) - Address of KRP if one is allocated
0488 995
0488 996 ROUTINE VALUE:
0488 997
0488 998 none
0488 999
0488 1000 SIDE EFFECTS:
0488 1001
0488 1002 none
0488 1003
0488 1004
0488 1005 DVI_DO_ITEM:
0488 1006 MOVL CURRENT_UCB(FP),R6 ; Get current UCB address
0488 1007 ASHL #1,R9,R1 ; Item index
0488 1008 MOVL W^DVI_ITEM_TABLE[R1],R0 ; Fetch associated item value
0488 1009 BGEQ 20$ ; Branch if not a special item
0488 1010 JMP (R0) ; Handle special items
0488 1011 20$: BBC #DVI_V_DEVTYPE,R0,40$ ; Branch if no specific device type
0488 1012 CMPB UCBSB_DEVCLASS(R6),#DCS_DISK ; Disk only item, is it a disk?
0488 1013 BEQL 40$ ; Branch if not, null item
0488 1014 BRW EXESDVI_NULL_ITEM
0488 1015 40$: EXTZV #DVI_V_STRUCT,#DVI_S_STRUCT,R0,R1 ; Get structure code
0488 1016 CASE R1,<=
```

```
51 56 E8 AD D0 0488 1006
51 59 FF 8F 78 0488 1007
50 FB79 CF41 D0 0491 1008
02 18 0497 1009
60 17 0499 1010
09 50 19 E1 049B 1011 20$:
01 40 A6 91 049F 1012
03 13 04A3 1013
008B 31 04A5 1014
51 50 03 13 EF 04A8 1015 40$:
04AD 1016
```



```
04AD 1017 DVI_UCB,- : UCB
04AD 1018 DVI_DDB,- : DDB
04AD 1019 DVI_VCB_RVT_AQB,- : VCB
04AD 1020 DVI_VCB_RVT_AQB,- : RVT
04AD 1021 DVI_VCB_RVT_AQB,- : AQB
04AD 1022 DVI_ORB,- : ORB
04AD 1023 >
04BD 1024 :
04BD 1025 : Fall through for VCB, RVT, or AQB
04BD 1026 :
04BD 1027 DVI_VCB_RVT_AQB:
55 EC AD D0 04BD 1028 MOVCL CURRENT_VCB(FP),R5 : Get VCB address if any
03 12 04C1 1029 BNEQ 45$ : Branch if none
006D 31 04C3 1030 BRW EXE$DVI_NULL_ITEM
04C6 1031
04C6 1032 45$: ASSUME DVI_C_RVT EQ DVI_C_VCB+1
04C6 1033 ASSUME DVI_C_AQB EQ DVI_C_VCB+2
51 03 C2 04C6 1034 SUBL #DVI_C_RVT,R1 : -1 = VCB, 0 = RVT, 1 = AQB
39 19 04C9 1035 BLSS DVI_STRUCT : Branch if VCB
20 14 04CB 1036 BGTR DVI_AQB : Branch if AQB
04CD 1037 :
04CD 1038 : Get Relative Volume Table Address if any
04CD 1039 :
025C 30 04CD 1040 BSBW DVI_GET_RVT : Get relative volume table adr
06 13 04D0 1041 BEQL DVI_NO_RVT : Branch if not a volume set
54 0B A3 9A 04D2 1042 MOVZBL RVT$B RVOLS(R3),R4 : Number of volumes in volume set
06 11 04D6 1043 BRB DVI_RVT
04D8 1044 DVI_NO_RVT:
52 01 D0 04D8 1045 MOVL #1,R2 : This is volume 1 of single volume set
54 01 D0 04DB 1046 MOVL #1,R4 : This is a single volume set
04DE 1047 DVI_RVT:
51 50 0A 00 EF 04DE 1048 EXTZV #DVI_V_OFFSET,#DVI_S_OFFSET,R0,R1 : Offset is RVT item
04E3 1049 :
04E3 1050 : R2 = volume number for this volume, 1 if not a volume set
04E3 1051 : R3 = RVT address or 0 if not a volume set
04E3 1052 : R4 = volume count or 1 if not a volume set
04E3 1053 :
04E3 1054 CASE R1,<-
04E3 1055 RVT_VOLCNT,- : VOLCNT - Number of volumes in the vol set
04E3 1056 RVT_ROOTDEVNAM,- : ROOTDEVNAM - Device name for root vol in s
04E3 1057 RVT_NEXTDEVNAM,- : NEXTDEVNAM - Next device name in vol set
04E3 1058 >
04ED 1059 :
04ED 1060 : Get ACP queue header block address - AQB
04ED 1061 :
04ED 1062 DVI_AQB:
55 10 A5 D0 04ED 1063 MOVL VCB$A AQB(R5),R5 : Get AQB address
11 19 04F1 1064 BLSS DVI_STRUCT : Branch if system space address
3E 11 04F3 1065 BRB EXE$DVI_NULL_ITEM : No AQB, no item data to return
04F5 1066 DVI_ORB:
55 1C A6 D0 04F5 1067 MOVL UCB$A ORB(R6),R5 : Get ORB address
09 11 04F9 1068 BRB DVI_STRUCT
04FB 1069 DVI_DDB:
55 28 A6 D0 04FB 1070 MOVL UCB$A DDB(R6),R5 : Get DDB address
03 11 04FF 1071 BRB DVI_STRUCT
0501 1072 DVI_UCB:
55 56 D0 0501 1073 MOVL R6,R5 : Get UCB address
```

```
0504 1074 : R5 = Address of structure containing desired field
0504 1075 :
0504 1076 :
0504 1077 DVI_STRUCT:
51 50 0A 00 EF 0504 1078 EXTZV #DVI_V_OFFSET,#DVI_S_OFFSET,R0,R1 ; Structure offset
51 50 55 51 CO 0509 1079 ADDL R1,R5 ; Source address of item to move
51 50 03 16 EF 050C 1080 EXTZV #DVI_V_DATATYPE,#DVI_S_DATATYPE,R0,R1 ; Data type
0511 1081 CASE R1,<=
0511 1082 EXESDVI_VALUE, - ; VALUE - move specified bytcnt
0511 1083 EXESDVI_CSTRING, - ; CSTRING - move the ascic string
0511 1084 DVI_BOOLEAN - ; BOOLEAN - test the bit
0511 1085 >
16 11 051B 1086 BRB EXESDVI_NULL_ITEM ; Out of range DTYPE
051D 1087 : Boolean data type
051D 1088 :
051D 1089 :
051D 1090 DVI_BOOLEAN:
EO AD 51 50 05 1A EF 051D 1091 EXTZV #DVI_V_POSIT,#DVI_S_POSIT,R0,R1 ; Bit position
55 01 51 EF 0522 1092 EXTZV R1,#T,(R5),SCRATCH(FP) ; Get the bit and save it
EO AD DE 0528 1093 MOVAL SCRATCH(FP),R5 ; Point to the saved bit
07 11 052C 1094 BRB EXESDVI_VALUE
052E 1095 :
052E 1096 : Counted string data type
052E 1097 :
052E 1098 EXESDVI_CSTRING::
54 85 9A 052E 1099 MOVZBL (R5)+,R4 ; Get size of string, advance adr
07 11 0531 1100 BRB EXESDVI_MOVE_ITEM
0533 1101 : Null item to return to user
0533 1102 :
0533 1103 :
0533 1104 EXESDVI_NULL_ITEM::
50 D4 0533 1105 CLRL R0 ; Set size field to 0
54 50 09 0A EF 0535 1106 EXESDVI_VALUE::
0535 1107 EXTZV #DVI_V_BYTCNT,#DVI_S_BYTCNT,R0,R4 ; Size of item to move
053A 1108 :
053A 1109 : R4 = size of item to move in bytes
053A 1110 : R5 = source address to move from
053A 1111 : R7 = Destination address - already probed
053A 1112 : R8 = Address to return length or 0 - already probed
053A 1113 : R10 = Size of return buffer for item, zero fill this buffer
053A 1114 :
053A 1115 EXESDVI_MOVE_ITEM::
54 5A D1 053A 1116 CMPL R10,R4 ; If user buffer is too small
03 18 053D 1117 BGEQ 10$
54 5A D0 053F 1118 MOVL R10,R4 ; Move as much as will fit
58 D5 0542 1119 10$: TSTL R8 ; Return length requested?
03 13 0544 1120 BEQL 20$ ; Branch if not
68 54 B0 0546 1121 MOVW R4,(R8) ; Set size of data returned
67 5A 00 65 54 2C 0549 1122 20$: MOVCS R4,(R5),#0,R10,(R7) ; Store item zero filled
05 05 054F 1123 RSB
```

```
0550 1125 .SBTTL Special Items
0550 1126
0550 1127 : CONCEALED - return boolean indicating whether device is concealed
0550 1128
0550 1129 SPC_CONCEALED:
0550 1130 CLRL SCRATCH(FP) : Will hold bit to indicate concealed
0553 1131 CMPW #SS$ CONCEALED, STATUS(FP) : Is it actually concealed?
0559 1132 BNEQ 15$ : NEQ means answer is false
055B 1133 INCL SCRATCH(FP) : Set answer to true
055E 1134 15$: MOVL #1, R4 : Set length of data to move
0561 1135 MOVAL SCRATCH(FP), R5 : Point to data
0565 1136 BRB EXESDVI_MOVE_ITEM
0567 1137
0567 1138 : VOLNUMBER - return relative volume number
0567 1139
0567 1140 SPC_VOLNUMBER:
0567 1141 MOVL CURRENT_VCB(FP), R5 : If not mounted,
056B 1142 BEQL EXESDVI_NULL_ITEM : Then return zero
056D 1143 MOVZWL VCB$W RVN(R5), R4 : Fetch RVN field
0571 1144 BNEQ EXESDVI_VALUE_IN_R4 : Non-zero if in a vol set
0573 1145 INCL R4 : It should really be vol 1
0575 1146
0575 1147 : ***** Fall through to EXESDVI_VALUE_IN_R4
0575 1148
0575 1149 RVT items - VOLCNT, ROOTDEVNAM, NXTDEVNAM
0575 1150
0575 1151 : R4 = Number of volumes in volume set, 1 if not a volume set
0575 1152
0575 1153 RVT_VOLCNT:
0575 1154
0575 1155 : R4 = long word value to return to caller
0575 1156
0575 1157 EXESDVI_VALUE_IN_R4::
0575 1158 MOVAL SCRATCH(FP), R5 : Address to store VOLCNT
0579 1159 MOVL R4, (R5) : Save the volume count
057C 1160 MOVL #4, R4 : Number of bytes to return
057F 1161 BRB EXESDVI_MOVE_ITEM
0581 1162
0581 1163 : PID - Convert internal PID in UCB to extended PID for return
0581 1164
0581 1165 SPC_PID:
0581 1166 MOVL UCB$L PID(R6), R0 : Internal PID into R0
0585 1167 CVTPID: JSB EXESIPID_TO_EPID : Convert to extended
058B 1168 PUT4: MOVL R0, R4 : Put value in register 4
058E 1169 BRB EXESDVI_VALUE_IN_R4 : Join the common code
0590 1170
0590 1171 : ACPPID - Convert internal PID in AQB to extended PID for return
0590 1172
0590 1173 SPC_ACPPID:
0590 1174 MOVL CURRENT_VCB(FP), R0 : R0 -> volume control block
0594 1175 BEQL PUT4 : Return null item if zero
0596 1176 MOVL VCB$L_AQB(R0), R0 : Now R0 -> ACP Queue Block
059A 1177 BEQL PUT4 : Return null item if zero
059C 1178 MOVL AQB$L ACPPID(R0), R0 : Now R0 has the internal pid
05A0 1179 BRB CVTPID : Convert the pid and return
05A2 1180
05A2 1181 : R2 = Volume number of this volume, 1 if not a volume set
```



```
05A2 1182 : R3 = RVT address, 0 if not a volume set
05A2 1183 : R4 = Volume count, 1 if not a volume set
05A2 1184
05A2 1185 RVT_ROOTDEVNAM:
53 D5 05A2 1186 TSTL R3 : If not a volume set
31 13 05A4 1187 BEQL SPC_DEVNAM : Return this volume's device name
52 D4 05A6 1188 CLRL R2 : Otherwise return devnam for first vol
07 11 05A8 1189 RVT_NEXTDEVNAM:
56 40 A342 D0 05AA 1190 BRB 20$ : Loop 0 or more times
05AF 1191 10$: MOVL RVTSL_UCBLST-4(R3)[R2],R6 : Get UCB for this RVN
26 19 05AF 1192 : RVN is base 1, table is base 0
F5 52 54 F3 05B1 1193 20$: BLSS SPC_DEVNAM : Branch if UCB present
FF7B 31 05B5 1194 20$: AOBLEQ R4,R2,10$ : Try next RVN
05B5 1195 DVI_NULL_ITEM_1:
05B5 1196 BRW EXESDVI_NULL_ITEM
05B8 1197
05B8 1198 : Device Name String - DEVNAM
05B8 1199
05B8 1200 SPC_ALLDEVNAM:
54 01 D0 05B8 1201 MOVL #1,R4 : flag IOC$CVT_DEVNAM to return the
1D 11 05B8 1202 BRB SPC2 : allocation class + device name
05BD 1203
05BD 1204 SPC_FULLDEVNAM:
54 D4 05BD 1205 CLRL R4 : flag IOC$CVT_DEVNAM to return the
19 11 05BF 1206 BRB SPC2 : fully qualified device name
05C1 1207
05C1 1208 SPC_TT_PHYDEVNAM:
EF 38 A6 02 E1 05C1 1209 BBC #DEV$V_TRM,UCB$$_DEVCHAR(R6),DVI_NULL_ITEM_1 : Non-terminal?
EA 38 A6 0D E0 05C6 1210 BBS #DEV$V_NET,UCB$$_DEVCHAR(R6),DVI_NULL_ITEM_1 : Network dev?
07 3C A6 02 E0 05CB 1211 BBS #DEV$V_RTT,UCB$$_DEVCHAR2(R6),SPC_DEVNAM : Skip remote term's
56 00A0 C6 D0 05D0 1212 MOVL UCB$$_TL_PHYUCB(R6),R6 : Fetch physical UCB from virtual
DE 13 05D5 1213 BEQL DVI_NULL_ITEM_1 : None, go return null string
05D7 1214 SPC_DEVNAM:
54 01 CE 05D7 1215 MNEGL #1,R4 : Force nodename to be left off
55 56 D0 05DA 1216 SPC2: MOVL R6,R5 : UCB Address
50 5A D0 05DD 1217 MOVL R10,R0 : Size of return buffer
51 57 D0 05E0 1218 MOVL R7,R1 : Address of return buffer - pre probed
00000000'GF 16 05E3 1219 JSB G*IOC$CVT_DEVNAM : Get device name "ddcu:"
54 51 D0 05E9 1220 MOVL R1,R4 : Size of string returned
55 57 D0 05EC 1221 MOVL R7,R5 : Address of string
FF48 31 05EF 1222 BRW EXESDVI_MOVE_ITEM : Move to self, zero filling.
05F2 1223
05F2 1224 : Device lock name
05F2 1225
05F2 1226 LCK_FOR:
C1 3C A6 00 E0 05F2 1227 BBS S*#DEV$V_CLU,UCB$$_DEVCHAR2(R6),SPC_ALLDEVNAM : Cluster-visible?
C4 11 05F7 1228 BRB SPC_FULLDEVNAM : Not cluster-visible
05F9 1229 SPC_DEVLOCKNAM:
54 10 D0 05F9 1230 MOVL #16, R4 : OUTLEN is 16 bytes
56 EB AD D0 05FC 1231 MOVL CURRENT_UCB(FP),R6 : Setup for DVI_GET_RVT routine
55 EC AD D0 0600 1232 MOVL CURRENT_VCB(FP),R5
EC 13 0604 1233 BEQL LCK_FOR : EQL means not mounted
E7 38 A6 18 E0 0606 1234 BBS S*#DEV$V_FOR,UCB$$_DEVCHAR(R6),LCK_FOR : Foreign?
060B 1235
51 DC AD D0 060B 1236 MOVL KRP(FP),R1 : Retrieve allocated KRP
15 12 060F 1237 BNEQ 20$ : Continue if KRP has been allocated
51 00000000'GF 9E 0611 1238 MOVAB G*CTL$GL_KRPFL,R1 : Retrieve address of KRP queue listhead
```

```
51 04 B1 0F 0618 1239 REMQUE 04(R1),R1 : Retrieve KRP from lookaside list
      04 1C 061C 1240 BVC 10$ : Continue if got one
      061E 1241 BUG_CHECK KRPEMPTY,FATAL : Otherwise bugcheck
      0622 1242
DC AD 51 D0 0622 1243 10$: MOVL R1,KRP(FP) : Save address of KRP in local storage
      2C A6 D5 0626 1244 20$: TSTL UCBSL_PID(R6) : Is the device allocated?
      05 13 0629 1245 BEQL 30$ : EQL means it is not
      61 01 90 062B 1246 MOVB #DVI_K_PRIVATE,(R1) : Setup the prefix byte in KRP
      03 11 062E 1247 BRB 40$
      61 02 90 0630 1248 30$: MOVB #DVI_K_SHAREABLE,(R1) : Setup the prefix byte in KRP
      0DF6 30 0633 1249 40$: BSBW DVI_GET_RVT : Get relative volume table address
      09 12 0636 1250 BNEQ 50$ : NEQ means it is a volume set
55 00000080 8F C0 0638 1251 ADDL #VCBST_VOLCKNAM,R5 : Add in offset to 'name'
      04 11 063F 1252 BRB 60$
55 53 18 C1 0641 1253 50$: ADDL3 #RVTST_VLSLCKNAM,R3,R5 : Add in offset to 'name'
      01 A1 85 7D 0645 1254 60$: MOVQ (R5)+,T(R1) : Move the 12 bytes to the buffer
      09 A1 65 D0 0649 1255 MOVL (R5),6(R1)
      0D A1 D4 064D 1256 CLRL 13(R1) : Zero bytes 14-16
      55 61 DE 0650 1257 MOVAL (R1),R5 : Point to the whole buffer
      FEE4 31 0653 1258 BRW EXESDVI_MOVE_ITEM
      0656 1259 :
      0656 1260 : Volume set member
      0656 1261 :
      0656 1262 SPC_VOLSETMEM:
54 01 D0 0656 1263 MOVL #1,R4 : Boolean answer is one byte long
      E0 AD D4 0659 1264 CLRL SCRATCH(FP) : Assume not a volume set
56 E8 AD D0 065C 1265 MOVL CURRENT_UCB(FP),R6 : Setup for DVI_GET_RVT
55 EC AD D0 0660 1266 MOVL CURRENT_VCB(FP),R5
      08 13 0664 1267 BEQL 20$ : EQL means not mounted
      00C3 30 0666 1268 BSBW DVI_GET_RVT : Get relative volume table address
      03 13 0669 1269 BEQL 20$ : EQL means not a volume set
      E0 AD D6 066B 1270 INCL SCRATCH(FP) : Set volume set to True
55 E0 AD 9E 066E 1271 20$: MOVAB SCRATCH(FP),R5 : scratch(fp) is where the answer is
      FEC5 31 0672 1272 BRW EXESDVI_MOVE_ITEM
      0675 1273
      0675 1274 DVI_NULL_ITEM_2:
      FEBB 31 0675 1275 BRW EXESDVI_NULL_ITEM
      0678 1276 :
      0678 1277 : Volume name - strip trailing blanks
      0678 1278 :
      0678 1279 SPC_VOLNAM:
55 EC AD D0 0678 1280 MOVL CURRENT_VCB(FP),R5 : VCB address
      F7 13 067C 1281 BEQL DVI_NULL_ITEM_2 : No data if not mounted
      55 14 C0 067E 1282 ADDL #VCBST_VOLNAME,R5 : Address of volname string
      54 0B D0 0681 1283 MOVL #11,R4 : Base 0 count of characters in name
      20 6544 91 0684 1284 10$: CMPB (R5)[R4],#^A/ / : Strip off trailing blanks
      03 12 0688 1285 BNEQ 20$ : Branch if not a blank
      F7 54 F4 068A 1286 SOBGEQ R4,10$ : Try next character
      54 D6 068D 1287 20$: INCL R4 : Actual byte count
      FEAB 31 068F 1288 BRW EXESDVI_MOVE_ITEM : Go move the volume name
```

```
0692 1290 : LOGVOLNAM - logical volume name
0692 1291 :
0692 1292 :
0692 1293 : SPC_LOGVOLNAM:
51 DC AD D0 0692 1294 : MOVL KRP(FP),R1 : Retrieve allocated KRP
15 12 0696 1295 : BNEQ 5$ : Continue if KRP has been allocated
51 00000000'GF 9E 0698 1296 : MOVAB G^CTL$GL_KRPFL,R1 : Retrieve address of KRP queue listhead
51 04 B1 OF 069F 1297 : REMQUE @4(R1),RT : Retrieve KRP from lookaside list
04 1C 06A3 1298 : BVC 1$ : Continue if got one
06A5 1299 : BUG_CHECK KRPEMPTY,FATAL : Otherwise bugcheck
06A9 1300 :
DC AD 51 D0 06A9 1301 1$: MOVL R1,KRP(FP) : Save address of KRP in local storage
61 D4 06AD 1302 5$: CLRL (R1) : Set up a null LOGVOLNAM
55 EC AD D0 06AF 1303 : MOVL CURRENT_VCB(FP),R5 : If no volume control block
70 13 06B3 1304 : BEQL 80$ : then no logical volume name
0074 30 06B5 1305 : BSBW DVI_GET_RVT : Return RVT in R3 or zero
06 13 06B8 1306 : BEQL 10$ : Branch if not a volume set
56 44 A3 D0 06BA 1307 : MOVL RVT$LCB_LST(R3),R6 : Get Root UCB address
65 13 06BE 1308 : BEQL 80$ : Branch if no UCB, really an error
54 00000000'9F D0 06C0 1309 10$: MOVL @CTL$GL_PCB,R4 : Get the PCB address for this process
54 0080 C4 D0 06C7 1310 : MOVL PCB$L_JIB(R4),R4 : Get the JIB address
54 64 DE 06CC 1311 : MOVAL JIB$L_MTLFL(R4),R4 : Get the job-wide mount list head
06CF 1312 :
51 0B A5 02 06 EF 06CF 1313 : ASSUME VCB$V_GROUP EQ VCB$V_SYSTEM-1
06CF 1314 : EXTZV #VCB$V_GROUP,#2,VCB$B_STATUS(R5),R1 ; 0 = Process
06D5 1315 : : 1 = Group, 2 = System
0A 13 06D5 1316 : BEQL 20$ : Branch if not mounted /SYSTEM or /GROUP
54 00000000'EF DE 06D7 1317 : MOVAL IOC$GQ_MOUNTLST,R4 : Search System/Group Mounted Vol List
51 51 CE 06DE 1318 : MNEGL R1,R1 : System = -2, Group = -1
52 54 D0 06E1 1319 20$: MOVL R4,R2 : Copy list head address
06E4 1320 :
52 62 D0 06E4 1321 : ASSUME MTL$L_MTLFL EQ 0
54 52 D1 06E7 1322 30$: MOVL MTL$L_MTLFL(R2),R2 : Get next entry on list
39 13 06EA 1323 : CMPL R2,R4 : End of list?
0C A2 56 D1 06EC 1324 : BEQL 80$ : Branch if yes, no MTL, really error
F2 12 06F0 1325 : CMPL R6,MTL$L_UCB(R2) : MTL entry for this UCB?
53 D5 06F2 1326 : BNEQ 30$ : Try next if not
05 13 06F4 1327 : TSTL R3 : Volume set?
E9 0B A2 00 E1 06F6 1328 : BEQL 40$ : Branch if not, this MTL is it
06FB 1329 : BBC #MTL$V_VOLSET,MTL$B_STATUS(R2),30$ : Yes, get the right MTL entry
06FB 1330 :
06FB 1331 : R2 = Mounted Volume List (MTL) entry address
06FB 1332 :
54 00000000'9F D0 06FB 1333 :
54 DD 0702 1334 40$: MOVL @CTL$GL_PCB,R4 : PCB address for this process
00000000'EF 16 0704 1335 : PUSHL R4 : Save for unlock call
070A 1336 : JSB LNMSLOCKR : Lock Logical name Mutex for reading
070A 1337 :
070A 1338 : ***** Note that R2 is preserved across the above call
070A 1339 :
52 10 A2 D0 070A 1340 : MOVL MTL$L_LOGNAME(R2),R2 : Get logical name table entry adr
OC 13 070E 1341 : BEQL 50$ : Branch if none present
51 11 A2 9A 0710 1342 : MOVZBL LNMB$T_NAME(R2),R1 : Size of name
51 D6 0714 1343 : INCL R1 : Include count byte
DC BD 11 A2 51 28 0716 1344 : MOVC3 R1,LNMB$T_NAME(R2),@KRP(FP)
071C 1345 :
54 8ED0 071C 1346 50$: POPL R4 : Save logical name in KRP
: Get parameters for unlock call
```


- System Services to Get Device Informat 16-SEP-1984 02:14:35 VAX/VMS Macro V04-00 Page 27
Special Items 5-SEP-1984 03:53:32 [SYS.SRC]SYSGETDVI.MAR;1 (11)

```

00000000'EF 16 071F 1347 JSB LNMSUNLOCK ; Release Logical Name Table Mutex
55 DC AD D0 0725 1348 80$: MOVL KRP(FP),R5 ; Address of counted string to return
FE02 31 0729 1349 BRW EXESDVI_CSTRING ; Go move counted string

```

[illegible]

```
072C 1351 :  
072C 1352 : Functional Description:  
072C 1353 :  
072C 1354 :   Get Relative Volume Table address if any  
072C 1355 :  
072C 1356 : Calling Sequence:  
072C 1357 :  
072C 1358 :   BSBW   DVI_GET_RVT  
072C 1359 :  
072C 1360 : Inputs:  
072C 1361 :  
072C 1362 :   R5 = VCB address  
072C 1363 :   R6 = UCB address  
072C 1364 :  
072C 1365 : Outputs:  
072C 1366 :  
072C 1367 :   If this UCB is part of a volume set:  
072C 1368 :     Condition code Z is CLEAR  
072C 1369 :     R2 = RVN for this volume  
072C 1370 :     R3 = RVT address  
072C 1371 :     Other registers are preserved  
072C 1372 :  
072C 1373 :   If this UCB is NOT part of a volume set:  
072C 1374 :     Condition code Z is SET  
072C 1375 :     R3 = 0  
072C 1376 :     Other registers are preserved  
072C 1377 :  
072C 1378 : DVI_GET_RVT:  
072C 1379 :   CLRL   R3 ; Assume not a volume set  
072C 1380 :   BBS    S^#DEV$V FOR,UCB$$_DEVCHAR(R6),20$ ; If FOREIGN, not a volume set  
0733 1381 :   MOVZWL VCB$$_RVN(R5),R2 ; Relative volume number  
0737 1382 :   BEQL   20$ ; Branch if not a volume set  
0739 1383 :   MOVL   VCB$$_RVT(R5),R3 ; Fetch RVT address (still could be 0)  
073D 1384 :   RSB    20$ ; Return Z bit set if R3 = 0
```

OA	38	A6	53	D4	072C	1379	CLRL	R3	:	Assume not a volume set
	52	OE	A5	E0	072E	1380	BBS	S^#DEV\$V FOR,UCB\$\$_DEVCHAR(R6),20\$:	If FOREIGN, not a volume set
			04	3C	0733	1381	MOVZWL	VCB\$\$_RVN(R5),R2	:	Relative volume number
	53	20	A5	13	0737	1382	BEQL	20\$:	Branch if not a volume set
				D0	0739	1383	MOVL	VCB\$\$_RVT(R5),R3	:	Fetch RVT address (still could be 0)
				05	073D	1384	RSB	20\$:	Return Z bit set if R3 = 0

```
073E 1386 :  
073E 1387 : Volume free blocks  
073E 1388 :  
073E 1389 SPC_FREEBLOCKS:  
55 E0 AD DE 073E 1390 MOVAL SCRATCH(FP),R5 : Stash a pointer to the answer  
65 D4 0742 1391 CLRL (R5) : Assume zero blocks  
56 E8 AD D0 0744 1392 MOVL CURRENT_UCB(FP),R6 : Get the ucb address  
01 40 A6 91 0748 1393 CMPB UCBSB_DEVCLASS(R6),#DCS_DISK : Is it a disk?  
26 12 074C 1394 BNEQ 20$ : NEQ means not a disk, return 0 blocks  
56 EC AD D0 074E 1395 MOVL CURRENT_VCB(FP),R6 : Get the vcb for the disk  
20 13 0752 1396 BEQL 20$ : EQL means not mounted  
65 40 A6 D0 0754 1397 MOVL VCB$$_FREE(R6), (R5) : Assume that we will use free blocks from V  
54 10 A6 D0 0758 1398 MOVL VCB$$_AQB(R6),R4 : Get pointer to AQB  
16 13 075C 1399 BEQL 20$ : EQL means no AQB, use vcb$$_free (strange  
02 15 A4 91 075E 1400 CMPB AQB$$_ACPTYPE(R4),#AQB$$_F11V2 : Is it an ODS-2 ACP?  
10 12 0762 1401 BNEQ 20$ : NEQ means not ODS-2, use vcb$$_free  
54 7C A6 D0 0764 1402 MOVL VCB$$_VOLLKID(R6),R4 : Get the lock id for the volume lock  
0A 13 0768 1403 BEQL 20$ : EQL means no lock id, use vcb$$_free (shou  
7E 54 7D 076A 1404 MOVQ R4,-(SP) : Push the lock id and return value address  
0000077A'EF 02 FB 076D 1405 CALLS #2,EXESDVI_FREEBLOCKS : Call the routine to check the lock  
54 04 D0 0774 1406 20$: MOVL #4,R4 : If error, then vcb$$_free will be used  
FDC0 31 0777 1408 BRW EXESDVI_MOVE_ITEM : Answer is 4 bytes long  
077A 1409 : Go move the item  
077A 1410 :  
077A 1411 : EXESDVI_FREEBLOCKS  
077A 1412 : Procedure to fetch the correct free block count from the XQP's volume  
077A 1413 : lock block. This procedure is also called by SHOW DEVICE (SHODEVUTL.B32) DCL  
077A 1414 : command to fetch the correct free blocks for SHOW DEVICE displays.  
077A 1415 :  
077A 1416 : Input:  
077A 1417 : 4(AP) - Lockid to fetch value block  
077A 1418 : Output:  
077A 1419 : 8(AP) - Address to store freeblocks field  
077A 1420 : Routine value:  
077A 1421 : RC - Status from getlki call  
077A 1422 : -  
077A 1423 :  
077A 1424 : Offsets from frame pointer for scratch storage  
077A 1425 :  
FFFFFEE0 077A 1426 VALBLK = -32 : Address to return value block  
FFFFFEE4 077A 1427 FREEBL = VALBLK + 4 : Free blocks are in second longword of value block  
FFFFFEE0 077A 1428 ITMLST = -16 : Build 4 longword item list  
077A 1429 :  
0000 077A 1430 .ENTRY EXESDVI_FREEBLOCKS,0  
5E 20 C2 077C 1431 SUBL2 #32,SP : Get scratch area  
FO AD 02030010 8F D0 077F 1432 MOVL #<<LKIS VALBLK@16>!16>,ITMLST(FP) : Item code and buffer length  
F4 AD E0 AD 9E 0787 1433 MOVAB VALBLK(FP),ITMLST+4(FP) : Address of value block  
F8 AD 7C 078C 1434 CLRQ ITMLST+8(FP) : No return length, end of list  
078F 1435 $GETLKIW S EFN=S^#EXESC_SYSEFN, LKIDADR=4(AP), ITMLST=ITMLST(FP)  
08 BC 05 50 E9 07A4 1436 BLBC R0,10$ : Error? Then exit  
E4 AD D0 07A7 1437 MOVL FREEBL(FP),28(AP) : Send it back  
07AC 1438 10$: RET  
07AD 1439
```



```
07AD 1441
07AD 1442
07AD 1443
07AD 1444 : R6 = UCB address
07AD 1445 : R7 = Destination address - already probed
07AD 1446 : R8 = Address to return length or 0 - already probed
07AD 1447 : R10 = Size of return buffer for item, zero fill this buffer
07AD 1448
07AD 1449 SPC_MEDIA NAME:
01 40 A6 91 07AD 1450 CMPB UCB$B_DEVCLASS(R6), #DC$_DISK : If disk class OK
02 40 A6 91 07B1 1451 BEQL 10$
03 03 13 07B3 1452 CMPB UCB$B_DEVCLASS(R6), #DC$_TAPE : If tape class OK
04 03 13 07B7 1453 BEQL 10$
05 FD77 31 07B9 1454 5$: BRW EXE$DVI_NULL_ITEM : If not disk or tape return null
06 008C C6 D5 07BC 1455 10$: TSTL UCB$S_MEDIA_ID(R6) : Test to make sure field is not 0
07 F7 13 07C0 1456 BEQL 5$ : If 0 then return null item
08 51 DC AD D0 07C2 1457 MOVL KRP(FP), R1 : Get address of KRP to build
09 15 12 07C6 1458 BNEQ 30$ : the string (may be >4 chars)
10 51 00000000 GF 9E 07C8 1459 MOVAB G^CTL$GL KRPFL, R1 : Get 0 head
11 51 04 B1 0F 07CF 1460 REMQUE @4(R1), R1 : Get next KRP
12 04 1C 07D3 1461 BVC 20$ : Branch if not empty
13 07D5 1462 BUG CHECK KRPEMPTY, FATAL
14 DC AD 51 D0 07D9 1463 20$: MOVE R1, KRP(FP) : Save address of KRP address
15 54 D4 07DD 1464 30$: CLRL R4 : Init string length
16 50 008C C6 05 11 EF 07DF 1465 EXTZV #UCB$V_MEDIA_ID_N0, -
17 07E6 1466 #UCB$S_MEDIA_ID_N0, -
18 07E6 1467 UCB$S_MEDIA_ID(R6), -
19 07E6 1468 R0 : Extract character number
20 02 13 07E6 1469 BEQL 40$ : If zero null character
21 5F 10 07E8 1470 BSBB DVI_DECODE_MEDIA_CHAR : place ASCII char in string
22 50 008C C6 05 0C EF 07EA 1471 40$: EXTZV #UCB$V_MEDIA_ID_N1, -
23 07F1 1472 #UCB$S_MEDIA_ID_N1, -
24 07F1 1473 UCB$S_MEDIA_ID(R6), -
25 07F1 1474 R0 : Extract character number
26 02 13 07F1 1475 BEQL 50$ : If zero null character
27 54 10 07F3 1476 BSBB DVI_DECODE_MEDIA_CHAR : place ASCII char in string
28 50 008C C6 05 07 EF 07F5 1477 50$: EXTZV #UCB$V_MEDIA_ID_N2, -
29 07FC 1478 #UCB$S_MEDIA_ID_N2, -
30 07FC 1479 UCB$S_MEDIA_ID(R6), -
31 07FC 1480 R0 : Extract character number
32 02 13 07FC 1481 BEQL 60$ : If zero null character
33 49 10 07FE 1482 BSBB DVI_DECODE_MEDIA_CHAR : place ASCII char in string
34 50 008C C6 07 00 EF 0800 1483 60$: EXTZV #UCB$V_MEDIA_ID_NN, -
35 0807 1484 #UCB$S_MEDIA_ID_NN, -
36 0807 1485 UCB$S_MEDIA_ID(R6), -
37 0807 1486 R0 : value of two decimal
38 00000064 8F 50 D1 0807 1487 CMPL R0, #100 : digits
39 18 1F 080E 1488 BLSSU 70$ : If < 100 skip to tens
40 52 D4 0810 1489 CLRL R2 : Clear hundreds counter
41 50 64 8F 82 0812 1490 SUBB2 #100, R0 : Subtract 100 from value
42 04 19 0816 1491 BLSS 66$ : Branch if go negative
43 52 D6 0818 1492 INCL R2 : Incre tens counter
44 F6 11 081A 1493 BRB 65$ : Loop
45 50 64 8F 80 081C 1494 66$: ADDB2 #100, R0 : Get back to positive #
```

```
52 30 80 0820 1498 ADDDB2 #^X30, R2 ; Convert to ascii char
81 52 90 0823 1499 MOVVB R2, (R1)+ ; Move into string
54 D6 0826 1500 INCL R4 ; Adjust length
52 D4 0828 1501 70$: CLRL R2 ; Clear tens counter
50 0A 82 082A 1502 75$: SUBB2 #10, R0 ; Subtract 10 from value
04 19 082D 1503 BLSS 80$ ; Branch if go negative
52 D6 082F 1504 INCL R2 ; Incre tens counter
F7 11 0831 1505 BRB 75$ ; Loop
52 30 80 0833 1506 80$: ADDDB2 #^X30, R2 ; Convert to ascii char
81 52 90 0836 1507 MOVVB R2, (R1)+ ; Move into string
50 3A 80 0839 1508 ADDDB2 #^X3A, R0 ; Convert neg number to
; pos ascii character
81 50 90 083C 1509 MOVVB R0, (R1)+ ; Move it into the string
54 02 80 083F 1511 ADDDB2 #2, R4 ; Adjust the length
55 DC AD D0 0842 1512 MOVL KRP(FP), R5 ; Set source address
FCF1 31 0846 1513 BRW EXESDVI_MOVE_ITEM ; Move the string and process
; the next item
0849 1514
0849 1515
0849 1516 ;
0849 1517 ; R0 = character number
0849 1518 ; R1 = address of next free byte in string
0849 1519 ; R4 = length of string
0849 1520 ;
0849 1521 DVI_DECODE_MEDIA_CHAR:
0849 1522 CMPL R0, #26 ; Only 26 chars in alphabet
084C 1523 BGTRU 20$ ; If not 0-26 place "." in string
084E 1524 ADDL2 #^X40, R0 ; Convert number to ascii char
54 D6 0855 1525 10$: INCL R4 ; increment length
81 50 90 0857 1526 MOVVB R0, (R1)+ ; Move the char into string
05 085A 1527 RSB
50 2E 9A 085B 1528 20$: MOVZBL #^X2E, R0 ; Set "."
F5 11 085E 1529 BRB 10$
0860 1530
0860 1531 ;
0860 1532 ; R6 = UCB address
0860 1533 ; R7 = Destination address - already probed
0860 1534 ; R8 = Address to return length or 0 - already probed
0860 1535 ; R10 = Size of return buffer for item, zero fill this buffer
0860 1536 ;
0860 1537 SPC_MEDIA TYPE:
0860 1538 CMPB UCBSB_DEVCLASS(R6), #DCS_DISK ; If disk class OK
0864 1539 BEQL 10$
0866 1540 CMPB UCBSB_DEVCLASS(R6), #DCS_TAPE ; If tape class OK
086A 1541 BEQL 10$
086C 1542 BRW EXESDVI_NULL_ITEM ; If not disk or tape return null
51 E0 AD DE 086F 1543 10$: MOVAL SCRATCH(FP), R1 ; Set address to build string
54 D4 0873 1544 CLRL R4 ; Init char count
50 008C C6 05 1B EF 0875 1545 EXTZV #UCBSV_MEDIA_ID_T0, -
087C 1546 #UCBSS_MEDIA_ID_T0, -
087C 1547 UCBSL_MEDIA_ID(R6), -
087C 1548 R0 ; Extract character number
087C 1549 BEQL 20$ ; If zero null character
087E 1550 BSBB DVI_DECODE_MEDIA_CHAR ; place ASCII char in string
0880 1551 20$: EXTZV #UCBSV_MEDIA_ID_T1, -
50 008C C6 05 16 EF 0880 1552 #UCBSS_MEDIA_ID_T1, -
0887 1553 UCBSL_MEDIA_ID(R6), -
0887 1554
```

			0887	1555		R0	:	Extract character number
	02	13	0887	1556		30\$:	If zero null character
	BE	10	0889	1557		DVI_DECODE_MEDIA_CHAR	:	place ASCII char in string
			088B	1558	30\$:			
55	EO AD	DE	088B	1559		MOVAL	:	Set source address
	FCA8	31	088F	1560		BRW	:	Move the string and process
			0892	1561			:	the next item
			0892	1562				


```
0892 1564 .SBTTL Dual path and shadow set items
0892 1565 :
0892 1566 : Items for shadow sets
0892 1567 :
0892 1568 SPC_SHDW_CATCHUP_COPYING: ; Catchup copy in progress
0892 1569 SPC_SHDW_MERGE_COPYING: ; Merge copy in progress
0892 1570 SPC_SHDW_spare_bit_1: ; Just in case
0892 1571 SPC_SHDW_spare_bit_2: ; Just in case
50 000008E0'EF 9E 0892 1572 MOVAB EXESDVI_RETURN_FALSE, R0
OE 11 0899 1573 BRB MV_JUMP
089B 1574 SPC_SHDW_MASTER_NAME: ; Master name for set
089B 1575 SPC_SHDW_NEXT_MBR_NAME: ; Name of next member
089B 1576 SPC_SHDW_spare_string_1: ; Just in case
089B 1577 SPC_SHDW_spare_string_2: ; Just in case
50 FC94 CF 9E 089B 1578 MOVAB EXESDVI_NULL_ITEM, R0
07 11 08A0 1579 BRB MV_JUMP
08A2 1580 SPC_SHDW_spare_integer_1: ; Just in case
50 000008EC'EF 9E 08A2 1581 SPC_SHDW_spare_integer_2: ; Just in case
08A2 1582 MOVAB EXESDVI_RETURN_ZERO, R0
08A9 1583 ;fall through to MV_JUMP
08A9 1584 :
08A9 1585 : Since Shadow support is latent, we will jump into the mount verification
08A9 1586 : code in SYSLOA to process the item. This is a lot simpler than trying
08A9 1587 : to patch SYS at some future date.
08A9 1588 :
08A9 1589 MV_JUMP:
08A9 1590 JMP G^EXESMNTVER_DVI_ASSIST ; For now, this just does a JMP (R0)
08AF 1591 :
08AF 1592 :
08AF 1593 : DVIS_REMOTE_DEVICE - Device is served by a host other than the local VAX
08AF 1594 :
08AF 1595 SPC_REMOTE_DEVICE:
50 00000000'GF 9E 08AF 1596 MOVAB G^SCSSGA_LOCALSB, R0 ; Get the address of the local system block
56 E8 AD D0 08B6 1597 MOVL CURRENT_UCB(FP), R6 ; Get the address of the UCB
56 28 A6 D0 08BA 1598 MOVL UCBSL_DDB(R6), R6 ; Move down to the DDB
50 34 A6 D1 08BE 1599 CMPL DDBSL_SB(R6), R0 ; Compare DDB's SB with the local SB
1C 13 08C2 1600 BEQL EXESDVI_RETURN_FALSE ; EQL means that it is the local block
11 11 08C4 1601 BRB EXESDVI_RETURN_TRUE ; Set the flag, it is remote
08C6 1602 :
08C6 1603 :
08C6 1604 : DVIS_SHDW_MASTER - The device is really the "virtual" name for the shadow set
08C6 1605 :
08C6 1606 SPC_SHDW_MASTER:
11 56 EB AD D0 08C6 1607 MOVL CURRENT_UCB(FP), R6 ; Get the address of the UCB
3C A6 05 E1 08CA 1608 BBC #DEV$V_MSCP, - ; See if the mscp bit is set in the
08CF 1609 UCBSL_DEVCHAR2(R6), - ; second characteristics longword
08CF 1610 EXESDVI_RETURN_FALSE ; and return false if not set
00D4 C6 B5 08CF 1611 TSTW UCBSW_MSCPUNIT(R6) ; Unit # with high bit set is shadow master
0B 18 08D3 1612 BGEQ EXESDVI_RETURN_FALSE ; GEQ means that high bit is not set
00 11 08D5 1613 BRB EXESDVI_RETURN_TRUE ; Set the flag, it is the master
08D7 1614 :
08D7 1615 :
08D7 1616 : Routines to return specific values
08D7 1617 :
08D7 1618 EXESDVI_RETURN_TRUE:: ; Boolean TRUE
55 E0 AD DE 08D7 1619 MOVAL SCRATCH(FP), R5 ; Grab pointer to scratch area
65 01 D0 08DB 1620 MOVL #1, (R5) ; Return a one
```

06	11	08DE	1621	BRB	RETURN_TF	
		08E0	1622	EXESDVI_RETURN_FALSE::		; Boolean FALSE
55	E0 AD	DE	08E0	1623	MOVAL	SCRATCH(FP),R5 ; Grab pointer to scratch area
	65	D4	08E4	1624	CLRL	(R5) ; Return a zero
			08E6	1625	RETURN_TF:	
54	01	D0	08E6	1626	MOVL	#1,R4 ; Booleans are one byte long
	FC4E	31	08E9	1627	BRW	EXESDVI_MOVE_ITEM
			08EC	1628	EXESDVI_RETURN_ZERO::	; Integer 0
	54	D4	08EC	1629	CLRL	R4 ; Set the zero
	FC84	31	08EE	1630	BRW	EXESDVI_VALUE_IN_R4

```
08F1 1632 :  
08F1 1633 : DVIS_HOST_AVAIL - Host for the primary path is available  
08F1 1634 : DVIS_ALT_HOST_AVAIL - Host for the secondary path is available  
08F1 1635 :  
08F1 1636 SPC_ALT_HOST_AVAIL:  
E6 56 E8 AD D0 08F1 1637 MOVL CURRENT_UCB(FP), R6 : Get the UCB address  
3C A6 04 E1 08F5 1638 BBC #DEV$V_2P, - : If the dual-port bit is  
08FA 1639 UCBSL_DEVCHAR2(R6), - : clear in the characteristics,  
E1 3C A6 05 E1 08FA 1640 EXESDVI_RETURN_FALSE : return a false  
08FA 1641 BBC #DEV$V_MSCP, - : If the MSCP device bit is  
08FF 1642 UCBSL_DEVCHAR2(R6), - : clear in the characteristics,  
DC 3C A6 03 E0 08FF 1643 EXESDVI_RETURN_FALSE : return a false  
08FF 1644 BBS #DEV$V_CDP, - : If the class driver path bit is  
56 00C0 C6 D0 0904 1645 UCBSL_DEVCHAR2(R6), - : set in the characteristics,  
D5 18 0904 1646 EXESDVI_RETURN_FALSE : return a false (no 2P CDDB for these)  
OE 11 0909 1647 MOVL UCBSL_2P_CDDB(R6), R6 : Get the CDDB address for the second path  
090B 1648 BGEQ EXESDVI_RETURN_FALSE : Extra paranoia (false if not system address  
090D 1649 BRB HOST_AVAIL : Join the common code  
090D 1650  
090D 1651 SPC_HOST_AVAIL:  
C1 56 E8 AD D0 090D 1652 MOVL CURRENT_UCB(FP), R6 : Get the UCB address  
3C A6 05 E1 0911 1653 BBC #DEV$V_MSCP, - : If the MSCP device bit is clear, then  
0916 1654 UCBSL_DEVCHAR2(R6), - : it is a local path  
0916 1655 EXESDVI_RETURN_TRUE : and always return true  
56 00BC C6 D0 0916 1656 MOVL UCBSL_CDDB(R6), R6 : Get the DDB address for the primary path  
C0 12 A6 07 E0 091B 1657 HOST_AVAIL:  
091B 1658 BBS #CDDBSV_NOCONN, - : If the NOCONNECTION bit is set  
0920 1659 CDDBSV_STATUS(R6), - : in the status, then it is not avail  
0920 1660 EXESDVI_RETURN_FALSE : and return a false  
B5 11 0920 1661 BRB EXESDVI_RETURN_TRUE : Set the flag, it is available
```



```
0922 1663 : DVI$_HOST_COUNT - Number of hosts serving the device (either 0 or 1)
0922 1664 :
0922 1665 :
0922 1666 SPC_HOST_COUNT:
0922 1667   MOVL #4, R4 : Four is length of integer items
55 54 04 DO 0922 1667   MOVAB SCRATCH(FP), R5 : Get the pointer to the scratch longword
55 E0 AD 9E 0922 1668   MOVL #1, (R5) : Assume that the device has one server
65 65 01 DO 0922 1669   MOVL CURRENT_UCB(FP), R6 : Get the address of the UCB
56 E8 AD DO 092C 1670   BBC #DEVSV_2P, - : See if the dual path bit is clear in the
02 3C A6 04 E1 0930 1671   UCB$$_DEVCHAR2(R6), 10$ : second characteristics longword
65 D6 0935 1672   INCL (R5) : Bump the flag, it has a second path
FC00 31 0937 1674 10$: BRW EXE$DVI_MOVE_ITEM : Go move it
093A 1675 :
093A 1676 DVI_NULL_ITEM_3:
093A 1677   BRW EXE$DVI_NULL_ITEM
093D 1678 :
093D 1679 :
093D 1680 : DVI$_HOST_NAME - Node name of the host for the primary path
093D 1681 : DVI$_ALT_HOST_NAME - Node name of the host for the secondary path
093D 1682 :
093D 1683 SPC_ALT_HOST_NAME:
093D 1684   MOVL CURRENT_UCB(FP), R6 : Get the UCB address
F4 3C A6 04 E1 0941 1685   BBC #DEVSV_2P, - : If the dual-port bit is not
0946 1686   UCB$$_DEVCHAR2(R6), - : set in the characteristics,
0946 1687   DVI_NULL_ITEM_3 : return a null string
56 00A0 C6 DO 0946 1688   MOVL UCB$$_2P_DDB(R6), R6 : Get the DDB address for the second path
08 11 0948 1689   BRB HOST_NAME : Join the common code
094D 1690 SPC_HOST_NAME:
56 E8 AD DO 094D 1691   MOVL CURRENT_UCB(FP), R6 : Get the UCB address
56 28 A6 DO 0951 1692   MOVL UCB$$_DDB(R6), R6 : Get the DDB address for the primary path
0955 1693 HOST_NAME:
56 34 A6 DO 0955 1694   MOVL DDB$$_SB(R6), R6 : Get the SB address
54 44 A6 9A 0959 1695   MOVZBL SB$$_NODENAME(R6), R4 : Pick up length of ASCII string
55 45 A6 9E 095D 1696   MOVAB SB$$_NODENAME+1(R6), R5 : Pick up address of ASCII string
FBD6 31 0961 1697   BRW EXE$DVI_MOVE_ITEM : Go move the item
0964 1698 :
0964 1699 :
0964 1700 : DVI$_HOST_TYPE - Type of node of the host for the primary path
0964 1701 : DVI$_ALT_HOST_TYPE - Type of node of the host for the secondary path
0964 1702 :
0964 1703 SPC_ALT_HOST_TYPE:
0964 1704   MOVL CURRENT_UCB(FP), R6 : Get the UCB address
CD 3C A6 04 E1 0968 1705   BBC #DEVSV_2P, - : If the dual-port bit is not
096D 1706   UCB$$_DEVCHAR2(R6), - : set in the characteristics,
096D 1707   DVI_NULL_ITEM_3 : return a null string
56 00A0 C6 DO 096D 1708   MOVL UCB$$_2P_DDB(R6), R6 : Get the DDB address for the second path
08 11 0972 1709   BRB HOST_TYPE : Join the common code
0974 1710 SPC_HOST_TYPE:
56 E8 AD DO 0974 1711   MOVL CURRENT_UCB(FP), R6 : Get the UCB address
56 28 A6 DO 0978 1712   MOVL UCB$$_DDB(R6), R6 : Get the DDB address for the primary path
097C 1713 HOST_TYPE:
56 34 A6 DO 097C 1714   MOVL DDB$$_SB(R6), R6 : Get the SB address
55 34 A6 9E 0980 1715   MOVAB SB$$_RWTYPEN(R6), R5 : Pick up address of padded string
65 04 20 3A 0984 1716   LOCC #A'-', #4, (R5) : Look for the first blank
54 04 50 C3 0988 1717   SUBL3 R0, #4, R4 : R0 contains number of blanks (or zero)
FBAB 31 098C 1718   BRW EXE$DVI_MOVE_ITEM : Go move the item
```

```
098F 1720 .SBTTL Get UCB from channel or device name
098F 1721
098F 1722 :
098F 1723 : FUNCTIONAL DESCRIPTION:
098F 1724 : Given either the channel or the device name string, return
098F 1725 : the primary UCB/VCB addresses and the secondary UCB/VCB addresses
098F 1726 :
098F 1727 : INPUTS:
098F 1728 : R0 = CHAN if entered at DVI_USE_CHAN
098F 1729 : = DEVNAM if entered at DVI_USE_DEVNAM
098F 1730 : Neither the descriptor nor the string have been probed
098F 1731 : R4 = Current Process PCB Address
098F 1732 :
098F 1733 : OUTPUTS:
098F 1734 : R0 = status
098F 1735 : STATUS(FP) = Returned success status from IOC$SEARCHDEV
098F 1736 : = $$$_NORMAL or $$$_CONCEALED
098F 1737 : Only returned when entered at DVI_USE_DEVNAM
098F 1738 : PRIMARY_UCB(FP) = Address of the primary UCB
098F 1739 : PRIMARY_VCB(FP) = Address of the primary VCB
098F 1740 : SECONDARY_UCB(FP) = Address of the secondary UCB
098F 1741 : SECONDARY_VCB(FP) = Address of the secondary VCB
098F 1742 :
098F 1743 :
098F 1744 :
098F 1745 : .ENABL LSB
098F 1746 :
098F 1747 : DVI_USE_CHAN:
00000000'GF 16 098F 1748 JSB G*IOC$VERIFYCHAN : Verify channel number
6A 50 E9 0995 1749 BLBC R0,60$ : Branch if error
61 DD 0998 1750 PUSHL CCB$$_UCB(R1) : Get UCB out of CCB
00000000'GF 16 099A 1751 JSB G*$_SCH$IOLOCKR : Lock I/O database for read access
DO AD D6 09A0 1752 INCL IOUNLOCK(FP) : Note that unlock is required
02 BA 09A3 1753 POPR #^M<R1> : Recover UCB address
37 11 09A5 1754 BRB 30$
50 0144 8F 3C 09A7 1755 10$: MOVZWL #$$$_IVDEVNAM,R0 : Invalid device name error
54 11 09AC 1756 BRB 60$
50 0C DO 09AE 1757 20$: MOVL #$$$_ACCVIO,R0 : Access violation
4F 11 09B1 1758 BRB 60$
098F 1759 : DVI_USE_DEVNAM:
51 50 DO 09B3 1760 MOVL R0,R1 : Device name string specified?
EF 13 09B6 1761 BEQL 10$ : Branch if not, IVDEVNAM
098F 1762 IFNORD #8,(R1),20$ : Branch if descriptor cannot be read
00000000'GF 16 09BE 1763 JSB G*$_SCH$IOLOCKR : Lock I/O database for read access
DO AD D6 09C4 1764 INCL IOUNLOCK(FP) : Note that unlock is required
09C7 1765 :
09C7 1766 : ***** Note that the device name string still must be probed
09C7 1767 :
00000000'GF 16 09C7 1768 JSB G*IOC$SEARCHDEV : Search for device
32 50 E9 09CD 1769 BLBC R0,60$ : If error, return status
DB AD 50 B0 09D0 1770 MOVW R0,STATUS(FP) : Save success status
09D4 1771 : $$$_NORMAL or $$$_CONCEALED
09D4 1772 :
05 3C A1 08 E1 09D4 1773 BBC S*$_DEV$V_RED,UCB$_DEVCHAR2(R1),30$; Skip if not redirected
09D9 1774 : physical terminal UCB
51 00C0 C1 DO 09D9 1775 MOVL UCB$_TT_LOGUCB(R1),R1 : redirect to associated logical tty UCB
09DE 1776
```

```
09DE 1777 :  
09DE 1778 : R1 = desired UCB  
09DE 1779 : If the device has an associated mail box and it is not spooled, then  
09DE 1780 : the UCB in the AMB field is the secondary device. If, however the  
09DE 1781 : device is spooled, the AMB field (intermediate device) is the primary  
09DE 1782 : device and the final destination device is the secondary.  
09DE 1783 :  
05 53 52 51 D0 09DE 1784 30$: MOVL R1,R2 : Assume primary = secondary  
53 60 A1 D0 09E1 1785 MOVL UCB$$_AMB(R1),R3 : Get associated mail box if any  
05 38 A1 0D 13 09E5 1786 BEQL 50$ : Branch if none  
52 06 E1 09E7 1787 BBC S^#DEV$V_SPL,UCB$$_DEVCHAR(R1),40$ : Branch if not spooled  
53 03 D0 09EC 1788 MOVL R3,R2 : Spooled dev, primary = AMB = intermed dev  
51 53 D0 09EF 1789 BRB 50$  
09F1 1790 40$: MOVL R3,R1 : Not spooled, secondary = AMB  
09F4 1791 :  
09F4 1792 : R2 = primary UCB  
09F4 1793 : R1 = secondary UCB  
09F4 1794 :  
50 F0 AD DE 09F4 1795 50$: MOVAL PRIMARY_UCB(FP),R0 : Address to store primary UCB/VCB  
09F8 1796 BSBB SET_UCB_VCB : Store UCB and VCB  
52 51 D0 09FA 1797 MOVL R1,R2 : Secondary UCB  
04 04 10 09FD 1798 BSBB SET_UCB_VCB : Store secondary UCB/VCB  
50 01 D0 09FF 1799 MOVL #SS$_NORMAL,R0 : Set success status  
05 0A02 1800 60$: RSB  
0A03 1801 :  
0A03 1802 : .DSABL LSB  
0A03 1803 :  
0A03 1804 : Store UCB and its associated VCB address if any  
0A03 1805 :  
0A03 1806 : Inputs:  
0A03 1807 :  
0A03 1808 : R2 = UCB address  
0A03 1809 : R0 = address to store UCB/VCB  
0A03 1810 :  
0A03 1811 : Outputs:  
0A03 1812 :  
0A03 1813 : R0 updated to next quad word  
0A03 1814 : R1,R2 preserved  
0A03 1815 : R3 altered  
0A03 1816 : other registers preserved  
0A03 1817 :  
0A03 1818 SET_UCB_VCB:  
04 38 A2 53 D4 0A03 1819 CLRL R3 : Assume volume not mounted  
53 34 A2 E1 0A05 1820 BBC S^#DEV$V_MNT,UCB$$_DEVCHAR(R2),10$ : Branch if not mounted  
80 52 7D 0A0A 1821 MOVL UCB$$_VCB(R2),R3 : Get VCB address  
05 0A0E 1822 10$: MOVQ R2,(R0)+ : Store UCB/VCB  
0A11 1823 RSB  
0A12 1824 :  
0A12 1825 : .END
```


SYSGETDVI
Symbol table

1 13
- System Services to Get Device Informat 16-SEP-1984 02:14:35 VAX/VMS Macro V04-00
5-SEP-1984 03:53:32 [SYS.SRC]SYSGETDVI.MAR;1

Page 39
(19)

SS11	= 00000001			DEVSV_SHR	= 00000010		
ACCVIO	= 0000034E	R	02	DEVSV_SPL	= 00000006		
ACCVIO_1	= 00000298	R	02	DEVSV_SQD	= 00000005		
AQBSB_ACPTYPE	= 00000015			DEVSV_SRV	= 00000007		
AQBSK_F11V1	= 00000001			DEVSV_SSM	= 00000006		
AQBSK_F11V2	= 00000002			DEVSV_SWL	= 00000019		
AQBSK_JNL	= 00000006			DEVSV_TRM	= 00000002		
AQBSK_MTA	= 00000003			DEVSV_WCK	= 0000001F		
AQBSK_NET	= 00000004			DEVNAM	= 0000000C		
AQBSK_REM	= 00000005			DEVTAB	= 00000000	R	02
AQBSL_ACPPID	= 0000000C			DIBSK_LENGTH	= 00000074		
ASTADR	= 00000018			DIBSL_MAXBLOCK	= 00000070		
ASTPRM	= 0000001C			DIBST_DEVNAME	= 00000024		
BUGS_KRPEMPTY	= *****	X	02	DIBSW_DEVNAMEOFF	= 0000000E		
CCBSL_UCB	= 00000000			DIBSW_VOLNAMEOFF	= 00000020		
CDDBSV_NOCONN	= 00000007			DIR...	= FFFFFFFF		
CDDBSW_STATUS	= 00000012			DVISC_ACP_F11V1	= 00000001		
CHAN	= 00000008			DVISC_ACP_F11V2	= 00000002		
CHAN_DEVNAM	= 00000004			DVISC_ACP_JNL	= 00000006		
CTLSGL_KRPFL	= *****	X	02	DVISC_ACP_MTA	= 00000003		
CTLSGL_PCB	= *****	X	02	DVISC_ACP_NET	= 00000004		
CURRENT_UCB	= FFFFFFFE8			DVISC_ACP_REM	= 00000005		
CURRENT_VCB	= FFFFFFFEC			DVIB_ACPPID	= 00000040		
CVTPID	= 00000585	R	02	DVIS_ACPTYPE	= 00000042		
DCS_DISK	= 00000001			DVIS_ALL	= 0000006C		
DCS_TAPE	= 00000002			DVIS_ALLDEVNAM	= 000000EC		
DDBSL_ALLOCLS	= 0000003C			DVIS_ALLOCLASS	= 000000F2		
DDBSL_SB	= 00000034			DVIS_ALT_HOST_AVAIL	= 000000F4		
DDBST_NAME	= 00000014			DVIS_ALT_HOST_NAME	= 000000F6		
DEVSV_2P	= 00000004			DVIS_ALT_HOST_TYPE	= 000000F8		
DEVSV_ALL	= 00000017			DVIS_AVL	= 00000062		
DEVSV_AVL	= 00000012			DVIS_CCL	= 00000048		
DEVSV_CCL	= 00000001			DVIS_CLUSTER	= 0000003A		
DEVSV_CDP	= 00000003			DVIS_CONCEALED	= 00000044		
DEVSV_CLU	= 00000000			DVIS_CYLINDERS	= 00000028		
DEVSV_DIR	= 00000003			DVIS_DEVBUSIZ	= 00000008		
DEVSV_DMT	= 00000015			DVIS_DEVCHAR	= 00000002		
DEVSV_DUA	= 0000000F			DVIS_DEVCHAR2	= 0C0000E6		
DEVSV_ELG	= 00000016			DVIS_DEVCLASS	= 00000004		
DEVSV_FOD	= 0000000E			DVIS_DEVDEPEND	= 0000000A		
DEVSV_FOR	= 00000018			DVIS_DEVDEPEND2	= 0000001C		
DEVSV_GEN	= 00000011			DVIS_DEVLOCKNAM	= 000000F0		
DEVSV_IDV	= 0000001A			DVIS_DEVNAM	= 00000020		
DEVSV_MBX	= 00000014			DVIS_DEVSTS	= 000000E4		
DEVSV_MHT	= 00000013			DVIS_DEVTYPE	= 00000006		
DEVSV_MSCP	= 00000005			DVIS_DIR	= 0000004C		
DEVSV_NET	= 0000000D			DVIS_DMT	= 00000068		
DEVSV_ODV	= 0000001B			DVIS_DUA	= 0000005C		
DEVSV_OPR	= 00000007			DVIS_ELG	= 0000006A		
DEVSV_RCK	= 0000001E			DVIS_ERRCNT	= 00000014		
DEVSV_RCT	= 00000008			DVIS_FOD	= 0000005A		
DEVSV_REC	= 00000000			DVIS_FOR	= 0000006E		
DEVSV_RED	= 00000008			DVIS_FREEBLOCKS	= 0000002A		
DEVSV_RND	= 0000001C			DVIS_FULLDEVNAM	= 000000E8		
DEVSV_RTM	= 0000001D			DVIS_GEN	= 00000060		
DEVSV_RTT	= 00000002			DVIS_HOST_AVAIL	= 000000FA		
DEVSV_SDI	= 00000004			DVIS_HOST_COUNT	= 000000FC		

SYSGETDVI
Symbol table

J 13
- System Services to Get Device Informat 16-SEP-1984 02:14:35 VAX/VMS Macro V04-00
5-SEP-1984 03:53:32 [SYS.SRC]SYSGETDVI.MAR;1

Page 40
(19)

DVIS_HOST_NAME = 000000FE
DVIS_HOST_TYPE = 00000100
DVIS_IDV = 00000072
DVIS_LOCKID = 000000EA
DVIS_LOGVOLNAM = 0000002C
DVIS_MAXBLOCK = 0000001A
DVIS_MAXFILES = 0000003C
DVIS_MBX = 00000066
DVIS_MEDIA_ID = 0000011A
DVIS_MEDIA_NAME = 00000116
DVIS_MEDIA_TYPE = 00000118
DVIS_MNT = 00000064
DVIS_MOUNTCNT = 00000038
DVIS_NET = 00000058
DVIS_NEXTDEVNAM = 00000034
DVIS_ODV = 00000074
DVIS_OPCNT = 00000016
DVIS_OPR = 00000054
DVIS_OWNUIC = 00000010
DVIS_PID = 0000000E
DVIS_RCK = 0000007A
DVIS_RCT = 00000056
DVIS_REC = 00000046
DVIS_RECSIZ = 00000018
DVIS_REFCNT = 0000001E
DVIS_REMOTE_DEVICE = 00000102
DVIS_RND = 00000076
DVIS_ROOTDEVNAM = 00000032
DVIS_RTM = 00000078
DVIS_SDI = 0000004E
DVIS_SECTORS = 00000024
DVIS_SERIALNUM = 0000003E
DVIS_SERVED_DEVICE = 00000104
DVIS_SHDW_CATCHUP_COPYING = 00000106
DVIS_SHDW_MASTER = 00000108
DVIS_SHDW_MASTER_NAME = 0000010A
DVIS_SHDW_MEMBER = 0000010C
DVIS_SHDW_MERGE_COPYING = 0000010E
DVIS_SHDW_NEXT_MBR_NAME = 00000110
DVIS_SHDW_SPARE_BIT_1 = 0000011C
DVIS_SHDW_SPARE_BIT_2 = 0000011E
DVIS_SHDW_SPARE_INTEGER_1 = 00000124
DVIS_SHDW_SPARE_INTEGER_2 = 00000126
DVIS_SHDW_SPARE_STRING_1 = 00000120
DVIS_SHDW_SPARE_STRING_2 = 00000122
DVIS_SHR = 0000005E
DVIS_SPL = 00000052
DVIS_SQD = 00000050
DVIS_STS = 000000E2
DVIS_SWL = 00000070
DVIS_TRACKS = 00000026
DVIS_TRANSCNT = 00000036
DVIS_TRM = 0000004A
DVIS_TT_ALTYPEAHD = 000000B8
DVIS_TT_ANSICRT = 000000D6
DVIS_TT_APP_KEYPAD = 000000D2
DVIS_TT_AUTOBAUD = 000000AE

DVIS_TT_AVO = 000000DC
DVIS_TT_BLOCK = 000000DA
DVIS_TT_BRDCSTMBX = 000000B4
DVIS_TT_CRFILL = 00000092
DVIS_TT_DCL_MAILBX = 000000BC
DVIS_TT_DECCRT = 000000E0
DVIS_TT_DECCRT2 = 00000114
DVIS_TT_DIALUP = 000000C4
DVIS_TT_DISCONNECT = 000000C8
DVIS_TT_DMA = 000000B6
DVIS_TT_DRCS = 000000CE
DVIS_TT_EDIT = 000000DE
DVIS_TT_EDITING = 000000BE
DVIS_TT_EIGHTBIT = 0000009A
DVIS_TT_ESCAPE = 00000084
DVIS_TT_FALLBACK = 000000C2
DVIS_TT_HALFDUP = 000000A4
DVIS_TT_HANGUP = 000000B0
DVIS_TT_HOSTSYNC = 00000086
DVIS_TT_INSERT = 000000C0
DVIS_TT_LFFILL = 00000094
DVIS_TT_LOCALECHO = 000000AC
DVIS_TT_LOWER = 0000008C
DVIS_TT_MBXDSABL = 0000009C
DVIS_TT_MECHFORM = 000000A2
DVIS_TT_MECHTAB = 0000008E
DVIS_TT_MODEM = 000000A6
DVIS_TT_MODHANGUP = 000000B2
DVIS_TT_NOBRDCST = 0000009E
DVIS_TT_NOECHO = 00000080
DVIS_TT_NOTYPEAHD = 00000082
DVIS_TT_OPER = 000000A8
DVIS_TT_PAGE = 000000AA
DVIS_TT_PASSALL = 0000007E
DVIS_TT_PASTHRU = 000000CA
DVIS_TT_PHYDEVNAM = 00000112
DVIS_TT_PRINTER = 000000D0
DVIS_TT_READSYNC = 000000A0
DVIS_TT_REGIS = 000000D8
DVIS_TT_REMOTE = 00000098
DVIS_TT_SCOPE = 00000096
DVIS_TT_SCRIPT = 0000008A
DVIS_TT_SECURE = 000000C6
DVIS_TT_SETSPEED = 000000BA
DVIS_TT_SIXEL = 000000CC
DVIS_TT_SYSPWD = 000000D4
DVIS_TT_TTSYNC = 00000088
DVIS_TT_WRAP = 00000090
DVIS_UNIT = 0000000C
DVIS_VOLCOUNT = 00000030
DVIS_VOLNAM = 00000022
DVIS_VOLNUMBER = 0000002E
DVIS_VOLSETMEM = 000000EE
DVIS_VPROT = 00000012
DVIS_WCK = 0000007C
DVI_ACCVIO = 000003B0 R 02
DVI_AQB = 000004ED R 02

SY
VO

SYSGETDVI
Symbol table

K 13
- System Services to Get Device Information
16-SEP-1984 02:14:35 VAX/VMS Macro V04-00
5-SEP-1984 03:53:32 [SYS.SRC]SYSGETDVI.MAR;1

Page 41
(19)

DVI_BADPARAM	000003BC	R	02	EXESDVI_MOVE_ITEM	0000053A	RG	02
DVI_BIT	= 00000020			EXESDVI_NULL_ITEM	00000533	RG	02
DVI_BOOLEAN	00000510	R	02	EXESDVI_RETURN_FALSE	000008E0	RG	02
DVI_COMPLETE	00000423	R	02	EXESDVI_RETURN_TRUE	000008D7	RG	02
DVI_C_ANY	= 00000000			EXESDVI_RETURN_ZERO	000008EC	RG	02
DVI_C_AQB	= 00000004			EXESDVI_VALUE	00000535	RG	02
DVI_C_BOOLEAN	= 00000002			EXESDVI_VALUE_IN_R4	00000575	RG	02
DVI_C_CSTRING	= 00000001			EXESGETCHN	00000000	RG	03
DVI_C_DDB	= 00000001			EXESGETDEV	0000000A	RG	03
DVI_C_DISK	= 00000001			EXESGETDVI	00000014	RG	03
DVI_C_NEXTDEVNAM	= 00000002			EXESIPID TO EPID	*****	X	02
DVI_C_ORB	= 00000005			EXESMNTVER_DVI_ASSIST	*****	X	02
DVI_C_ROOTDEVNAM	= 00000001			EXESPROBEW_DSC	*****	X	02
DVI_C_RVT	= 00000003			EXE_GETDEV	0000025F	R	02
DVI_C_UCB	= 00000000			EXE_GETDVI	00000370	R	02
DVI_C_VALUE	= 00000000			FILBUF	00000298	R	02
DVI_C_VCB	= 00000002			FREEBL	= FFFFFFFE4	R	02
DVI_C_VOLCOUNT	= 00000000			HOST_AVAIL	0000091B	R	02
DVI_DDB	000004FB	R	02	HOST_NAME	00000955	R	02
DVI_DECODE_MEDIA_CHAR	00000849	R	02	HOST_TYPE	0000097C	R	02
DVI_DO_ITEM	00000488	R	02	IOCSVT_DEVNAM	*****	X	02
DVI_ERROR	000003AE	R	02	IOCSGQ_MOUNTLST	*****	X	02
DVI_ERROR_1	0000041F	R	02	IOCSSEARCHDEV	*****	X	02
DVI_EXASTCM	000003B5	R	02	IOCSUNLOCK	*****	X	02
DVI_GET_RVT	0000072C	R	02	IOCSVERIFYCHAN	*****	X	02
DVI_ITEM_TABLE	0000000F	R	02	IOSB	= 00000014		
DVI_K_PRIVATE	= 00000001			IOUNLOCK	FFFFFFD0		
DVI_K_SHAREABLE	= 00000002			ITEM CODE	= 00000128		
DVI_NO_RVT	000004D8	R	02	ITMLST	= FFFFFFFF0		
DVI_NULL_ITEM_1	000005B5	R	02	JIBSL_MTLFL	= 00000000		
DVI_NULL_ITEM_2	00000675	R	02	KRP	FFFFFFDC		
DVI_NULL_ITEM_3	0000093A	R	02	LCK FOR	000005F2	R	02
DVI_ORB	000004F5	R	02	LKIS_VALBLK	= 00000203		
DVI_RVT	000004DE	R	02	LNMSLOCKR	*****	X	02
DVI_STRUCT	00000504	R	02	LNMSUNLOCK	*****	X	02
DVI_S_BYTCNT	= 00000009			LNMBST NAME	= 00000011		
DVI_S_DATATYPE	= 00000003			MAX_ITEM CODE	= 00000127		
DVI_S_DEVTYPE	= 00000001			MOVE NAME	00000352	R	02
DVI_S_OFFSET	= 0000000A			MTLSB_STATUS	= 0000000B		
DVI_S_POSIT	= 00000005			MTLSL_LOGNAME	= 00000010		
DVI_S_SPCFLG	= 00000001			MTLSL_MTLFL	= 00000000		
DVI_S_STRUCT	= 00000003			MTLSL_UCB	= 0000000C		
DVI_UCB	00000501	R	02	MTLSV_VOLSET	= 00000000		
DVI_USE_CHAN	0000098F	R	02	MV_JUMP	000008A9	R	02
DVI_USE_DEVNAM	000009B3	R	02	NUCARG	= 00000020		
DVI_VCB_RVT_AQB	000004BD	R	02	OFFVAL	= 0000008C		
DVI_V_BYTCNT	= 0000000A			ORBSL_OWNER	= 00000000		
DVI_V_DATATYPE	= 00000016			ORBSW_PROT	= 00000018		
DVI_V_DEVTYPE	= 00000019			PCBSL_JIB	= 00000080		
DVI_V_OFFSET	= 00000000			PCBSL_PID	= 00000060		
DVI_V_POSIT	= 0000001A			PCBSW_ASTCNT	= 00000038		
DVI_V_SPCFLG	= 0000001F			PRS_IPL	= 00000012		
DVI_V_STRUCT	= 00000013			PRIBUF	= 0000000C		
EFN	= 00000004			PRILEN	= 00000008		
EXESC_SYSEFN	*****	X	02	PRIMARY_UCB	FFFFFFF0		
EXESDVI_CSTRING	0000052E	RG	02	PRIMARY_VCB	= FFFFFFFF4		
EXESDVI_FREEBLOCKS	0000077A	RG	02	PSLSS_PRVMOD	= 00000002		

SY
VO

SYSGETDVI
Symbol table

L 13

- System Services to Get Device Informat 16-SEP-1984 02:14:35 VAX/VMS Macro V04-00
5-SEP-1984 03:53:32 [SYS.SRC]SYSGETDVI.MAR;1

Page 42
(19)

PSL\$V_PVPMOD	= 00000016			SPC_SHDW_SPARE_STRING_2	00000898	R	02
PUT4	00000588	R	02	SPC-TT_PRYDEVNAM	000005C1	RR	02
RETLEN	= FFFFFFFDA			SPC-VOLNAM	00000678	RR	02
RETLEN_ADR	FFFFFFFE4			SPC-VOLNUMBER	00000567	RR	02
RETURN_TF	000008E6	R	02	SPC-VOLSETMEM	00000656	R	02
RVT\$B_NVOLS	= 00000008			SS\$-ACCVIO	= 0000000C		
RVT\$B_UCBLST	= 00000044			SS\$-BADPARAM	= 00000014		
RVT\$T_VLSLCKNAM	= 00000018			SS\$-BUFFEROVF	= 00000061		
RVT_NEXTDEVNAM	000005A8	R	02	SS\$-CONCEALED	= 00000069		
RVT_ROOTDEVNAM	000005A2	R	02	SS\$-EXASTLM	= 00002A04		
RVT_VOLCNT	00000575	R	02	SS\$-IVDEVNAM	= 00000144		
SAVABS...	= FFFFFFFD0			SS\$-NORMAL	= 00000001		
SAVED_A\$ADR	FFFFFFFD4			STATUS	FFFFFFFD8		
SB\$T_RWTYPE	= 00000034			SY\$SDCLAST	*****	GX	02
SB\$T_NODENAME	= 00000044			SY\$SGETLKIW	*****	GX	02
SCDBUF	= 00000014			TT\$V_CRFILL	= 0000000A		
SCDLEN	= 00000010			TT\$V-EIGHTBIT	= 0000000F		
SCH\$CLREF	*****	X	02	TT\$V-ESCAPE	= 00000003		
SCH\$IOLOCKR	*****	X	02	TT\$V-HALFDUP	= 00000014		
SCH\$IOUNLOCK	*****	X	02	TT\$V-HOSTSYNC	= 00000004		
SCH\$POSTEF	*****	X	02	TT\$V_LFFILL	= 0000000B		
SCRATCH	FFFFFFFE0			TT\$V-LOWER	= 00000007		
SCRATCH_SIZE	FFFFFFFDC			TT\$V-MBXDSABL	= 00000010		
SCSSGA_LOCALSB	*****	X	02	TT\$V-MECHFORM	= 00000013		
SECONDARY_UCB	= FFFFFFFF8			TT\$V-MECHTAB	= 00000008		
SECONDARY_VCB	= FFFFFFFFC			TT\$V-MODEM	= 00000015		
SET_UCB_VCB	00000A03	R	02	TT\$V-NOBRDCST	= 00000011		
SPC2	000005DA	R	02	TT\$V-NOECHO	= 00000001		
SPC_ACPPID	00000590	R	02	TT\$V-NOTYPEAHD	= 00000002		
SPC-ALLDEVNAM	000005B8	R	02	TT\$V-OPER	= 00000016		
SPC-ALT_HOST_AVAIL	000008F1	R	02	TT\$V-PASSALL	= 00000000		
SPC-ALT_HOST_NAME	0000093D	R	02	TT\$V-READSYNC	= 00000012		
SPC-ALT_HOST_TYPE	00000964	R	02	TT\$V-REMOTE	= 0000000D		
SPC-CONCEALED	00000550	R	02	TT\$V-SCOPE	= 0000000C		
SPC-DEVLOCKNAM	000005F9	R	02	TT\$V-SCRIPT	= 00000006		
SPC-DEVNAM	000005D7	R	02	TT\$V-TTSYNC	= 00000005		
SPC-FREEBLOCKS	0000073E	R	02	TT\$V-WRAP	= 00000009		
SPC-FULLDEVNAM	000005BD	R	02	TT2\$V-ALTYPEAHD	= 00000007		
SPC-HOST_AVAIL	0000090D	R	02	TT2\$V-ANSICRT	= 00000018		
SPC-HOST_COUNT	00000922	R	02	TT2\$V-APP_KEYPAD	= 00000017		
SPC-HOST_NAME	0000094D	R	02	TT2\$V-AUTOBAUD	= 00000001		
SPC-HOST_TYPE	00000974	R	02	TT2\$V-AVO	= 0000001B		
SPC-LOGVOLNAM	00000692	R	02	TT2\$V-BLOCK	= 0000001A		
SPC-MEDIA_NAME	000007AD	R	02	TT2\$V-BRDCSTMBX	= 00000004		
SPC-MEDIA_TYPE	00000860	R	02	TT2\$V-DCL_MAILBX	= 00000009		
SPC-PID	00000581	R	02	TT2\$V-DECCRT	= 0000001D		
SPC-REMOTE_DEVICE	000008AF	R	02	TT2\$V-DECCRT2	= 0000001E		
SPC_SHDW_CATCHUP_COPYING	00000892	R	02	TT2\$V-DIALUP	= 0000000F		
SPC_SHDW_MASTER	000008C6	R	02	TT2\$V-DISCONNECT	= 00000011		
SPC_SHDW_MASTER_NAME	00000898	R	02	TT2\$V-DMA	= 00000006		
SPC_SHDW_MERGE_COPYING	00000892	R	02	TT2\$V-DRCS	= 00000015		
SPC_SHDW_NEXT_RBR_NAME	00000898	R	02	TT2\$V-EDIT	= 0000001C		
SPC_SHDW_SPARE_BIT_1	00000892	R	02	TT2\$V-EDITING	= 0000000C		
SPC_SHDW_SPARE_BIT_2	00000892	R	02	TT2\$V-FALLBACK	= 0000000E		
SPC_SHDW_SPARE_INTEGER_1	000008A2	R	02	TT2\$V-HANGUP	= 00000002		
SPC_SHDW_SPARE_INTEGER_2	000008A2	R	02	TT2\$V-INSERT	= 0000000D		
SPC_SHDW_SPARE_STRING_T	00000898	R	02	TT2\$V-LOCALECHO	= 00000000		

SYSGETDVI
Symbol table

M 13
- System Services to Get Device Informat 16-SEP-1984 02:14:35 VAX/VMS Macro V04-00
5-SEP-1984 03:53:32 [SYS.SRC]SYSGETDVI.MAR;1

Page 43
(19)

TT2\$V_MODHANGUP = 00000003
TT2\$V_PASTHRU = 00000012
TT2\$V_PRINTER = 00000016
TT2\$V_REGIS = 00000019
TT2\$V_SECURE = 00000010
TT2\$V_SETSPEED = 00000008
TT2\$V_SIXEL = 00000014
TT2\$V_SYSPWD = 00000013
UCB\$B_DEVCLASS = 00000040
UCB\$B_DEVTYP = 00000041
UCB\$B_SECTORS = 00000044
UCB\$B_TRACKS = 00000045
UCB\$B_2P_CDD = 000000C0
UCB\$B_2P_DDB = 000000A0
UCB\$B_AMB = 00000060
UCB\$B_CDD = 000000BC
UCB\$B_DDB = 00000028
UCB\$B_DEVCHAR = 00000038
UCB\$B_DEVCHAR2 = 0000003C
UCB\$B_DEVDEPEND = 00000044
UCB\$B_DEVDEPEND2 = 00000048
UCB\$B_LOCKID = 00000020
UCB\$B_MAXBLOCK = 000000B0
UCB\$B_MEDIA_ID = 0000008C
UCB\$B_OPCNT = 00000070
UCB\$B_ORB = 0000001C
UCB\$B_PID = 0000002C
UCB\$B_STS = 00000064
UCB\$B_TL_PHYUCB = 000000A0
UCB\$B_TT_LOGUCB = 000000C0
UCB\$B_VCB = 00000034
UCB\$S_MEDIA_ID_NO = 00000005
UCB\$S_MEDIA_ID_N1 = 00000005
UCB\$S_MEDIA_ID_N2 = 00000005
UCB\$S_MEDIA_ID_NN = 00000007
UCB\$S_MEDIA_ID_TO = 00000005
UCB\$S_MEDIA_ID_T1 = 00000005
UCB\$V_MEDIA_ID_NO = 00000011
UCB\$V_MEDIA_ID_N1 = 0000000C
UCB\$V_MEDIA_ID_N2 = 00000007
UCB\$V_MEDIA_ID_NN = 00000000
UCB\$V_MEDIA_ID_TO = 0000001B
UCB\$V_MEDIA_ID_T1 = 00000016
UCB\$W_CYLINDERS = 00000046
UCB\$W_DEVBUFSIZ = 00000042
UCB\$W_DEVSTS = 00000068
UCB\$W_ERRCNT = 00000082
UCB\$W_MSCPUNIT = 000000D4
UCB\$W_REFC = 0000005C
UCB\$W_UNIT = 00000054
VALBLR = FFFFFFFE0
VCB\$B_STATUS = 0000000B
VCB\$B_AQB = 00000010
VCB\$B_FREE = 00000040
VCB\$B_MAXFILES = 00000044
VCB\$B_RVT = 00000020
VCB\$B_SERIALNUM = 00000064

VCB\$B_VOLLKID = 0000007C
VCB\$B_VOLCKNAM = 00000080
VCB\$B_VOLNAME = 00000014
VCB\$B_GROUP = 00000006
VCB\$B_SYSTEM = 00000007
VCB\$B_CLUSTER = 0000003C
VCB\$B_MCOUNT = 0000004C
VCB\$B_RECORDSZ = 00000050
VCB\$B_RVN = 0000000E
VCB\$B_TRANS = 0000000C
XTYPE = 00000000

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes
ABS	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	FFFFFFFF0 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
YF\$\$\$SYSGETDVI	00000A12 (2578.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
Y\$EXEPAGED	00000019 (25.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.06	00:00:01.14
Command processing	118	00:00:00.66	00:00:05.49
Pass 1	784	00:00:43.14	00:01:53.05
Symbol table sort	0	00:00:04.43	00:00:06.65
Pass 2	329	00:00:08.27	00:00:26.52
Symbol table output	60	00:00:00.45	00:00:01.58
Psect synopsis output	2	00:00:00.02	00:00:00.28
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1324	00:00:57.05	00:02:34.74

The working set limit was 2550 pages.

217209 bytes (425 pages) of virtual memory were used to buffer the intermediate code.

There were 150 pages of symbol table space allocated to hold 2749 non-local and 85 local symbols.

1825 source lines were read in Pass 1, producing 33 object records in Pass 2.

72 pages of virtual memory were used to define 46 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
\$255\$DUA28:[SYSLIB]SYSBLDMLB.MLB;1	1
\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	19
\$255\$DUA28:[SYSLIB]STARLET.MLB;2	20
TOTALS (all libraries)	40

3321 GETS were required to define 40 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:SYSGETDVI/OBJ=OBJ\$:SYSGETDVI MSRC\$:SYSGETDVI/UPDATE=(ENH\$:SYSGETDVI)+EXECML\$/LIB+SYS\$LIBRARY:SYSBLDMLB/LIB

0384 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

